```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
from zipfile import ZipFile
```

```
pip install tensorflow==2.12
```

```
Requirement already satisfied: tensorflow==2.12 in /usr/local/lib/python3.10/dist-packages (2.12.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (24.3.25)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.68.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (3.12.1)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.4.30)
Requirement already satisfied: keras<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.12.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (18.1.1)
Requirement already satisfied: numpy<1.24,>=1.22 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/py
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.16.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.12.3)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.1
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow==2
Requirement already satisfied: jaxlib<=0.4.30,>=0.4.27 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow==2.1
Requirement already satisfied: ml-dtypes>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow==2.12) (0.4
Requirement already satisfied: scipy>=1.9 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow==2.12) (1.13.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensc
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow=
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorf
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow=
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensor
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorb
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.1
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,>
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.13,
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-a
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-aut
```

```
import tensorflow as tf
print(tf.__version__)

import keras
print(keras.__version__)
```

```
2.12.0
2.12.0
```

```
!pip install tensorflow-addons
```

```
Requirement already satisfied: tensorflow-addons in /usr/local/lib/python3.10/dist-packages (0.23.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow-addons) (24.2)
Requirement already satisfied: typeguard<3.0.0,>=2.7 in /usr/local/lib/python3.10/dist-packages (from tensorflow-addons) (2.13.3)
```

```
from tensorflow.keras import layers, Input, Model, Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Concatenate, Dense, Dropout, Flatten, SeparableConv2D, BatchNormalization, Gl
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.applications import VGG19, InceptionV3
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.utils import plot_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
import tensorflow as tf
print(tf.__version__)
```

```
2.12.0
```

```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as img
#IMPORTING LIBRARIES
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt

import cv2
import itertools
import pathlib
import warnings
from PIL import Image
from random import randint
warnings.filterwarnings('ignore')

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.metrics import matthews_corrcoef as MCC
from sklearn.metrics import balanced_accuracy_score as BAS
from sklearn.metrics import classification_report, confusion_matrix

#   DataGenerator to read images and rescale images
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow import keras
from keras import layers
import tensorflow as tf
import tensorflow_addons as tfa

#   count each class samples
from collections import Counter

from tensorflow.keras.applications import VGG19
from tensorflow.keras.preprocessing import image_dataset_from_directory
from keras.utils.vis_utils import plot_model
from tensorflow.keras import Input
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import Conv2D, Flatten
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG
from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, GlobalAveragePooling2D

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Concatenate
from tensorflow.keras.models import Sequential


from distutils.dir_util import copy_tree, remove_tree

#   SMOTETomek from imblance library
from imblearn.combine import SMOTETomek

import os
import tensorflow as tf

# List only directories in the specified path
directory_path = "/content/drive/MyDrive/archive"
folders = [f for f in os.listdir(directory_path) if os.path.isdir(os.path.join(directory_path, f))]

print("Folders in '/content/drive/MyDrive/archive':")
print(folders)

# Print the TensorFlow version
print("TensorFlow Version:", tf.__version__)
```

```
Folders in '/content/drive/MyDrive/archive':
['Cercospora', 'Healthy', 'Leaf_rust', 'Miner', 'Phoma']
TensorFlow Version: 2.12.0
```

```
import tensorflow as tf
print(tf.__version__)
```

```
2.12.0
```

```
import tensorflow
from tensorflow import keras
from keras.models import Sequential,load_model,Model
from keras.layers import Conv2D,MaxPool2D,AveragePooling2D,Dense,Flatten,ZeroPadding2D,BatchNormalization,Activation,Add,Input,Dropout,(
from keras.optimizers import SGD
from keras.initializers import glorot_uniform
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint,EarlyStopping,ReduceLROnPlateau
```

```
X, y = [], []

## Images rescaling
datagen = ImageDataGenerator(rescale=1.0/255.0)

#   Load images by resizing and shuffling randomly
train_dataset = datagen.flow_from_directory("/content/drive/MyDrive/archive", target_size=(128, 128),batch_size=7000, shuffle=True)

### Seperate Dataset from  Data Genrator
X, y = train_dataset.next()
```

```
Found 58550 images belonging to 5 classes.
```

```
samples_before = len(X)
print("Images shape :\t", X.shape)
print("Labels shape :\t", y.shape)
```

```
Images shape :   (7000, 128, 128, 3)
Labels shape :   (7000, 5)
```

```
#   Number of samples in classes
print("Number of samples in each class:\t", sorted(Counter(np.argmax(y, axis=1)).items()))

#   class labels as per indices
print("Classes Names according to index:\t", train_dataset.class_indices)
```

```
Number of samples in each class:         [(0, 922), (1, 2303), (2, 966), (3, 2042), (4, 767)]
Classes Names according to index:        {'Cercospora': 0, 'Healthy': 1, 'Leaf_rust': 2, 'Miner': 3, 'Phoma': 4}
```

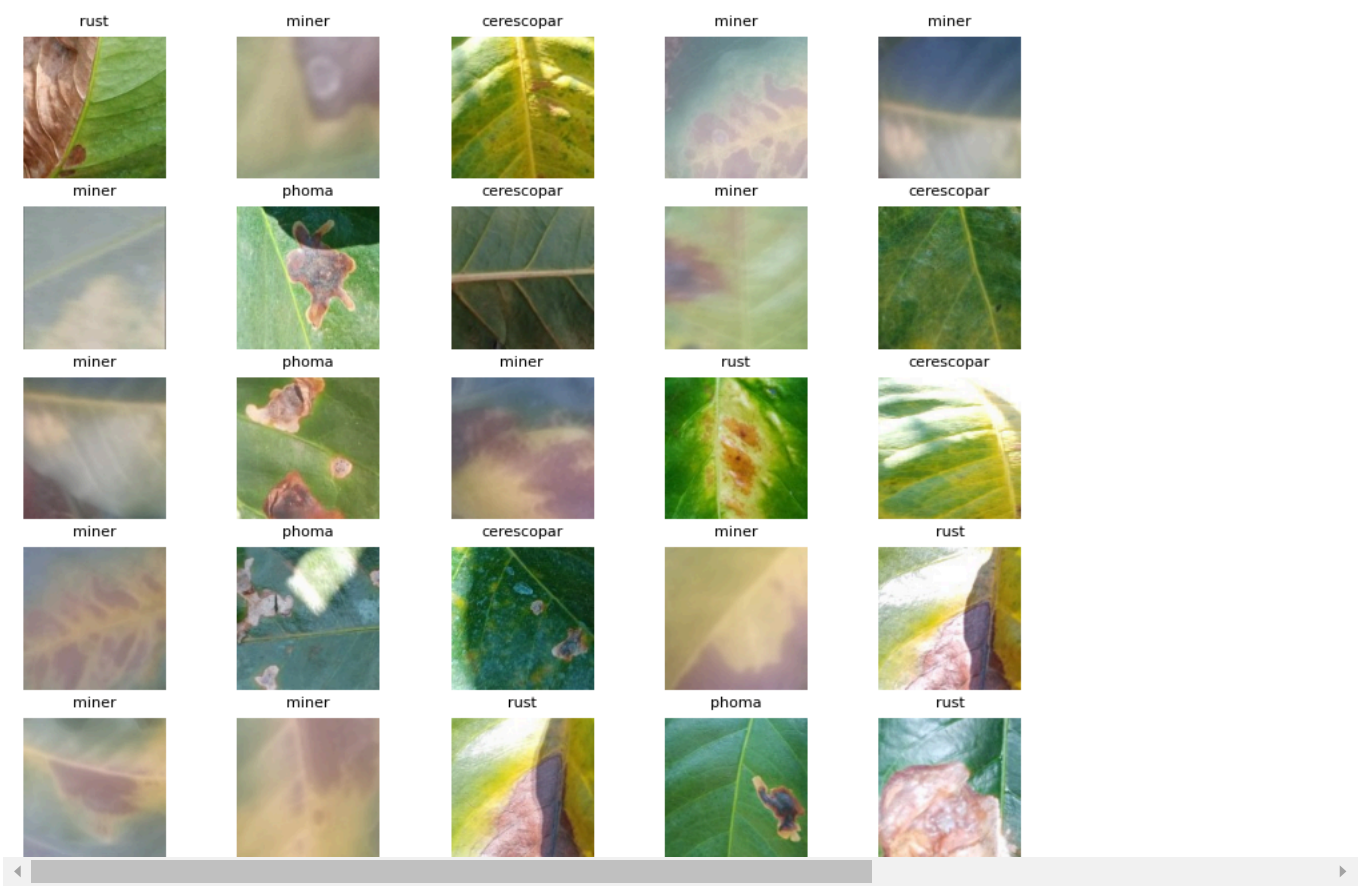
```
##random figures.
```

```
#   show some samples from the dataset randomly
import random
fig = plt.figure(figsize=(10,8))

CLASSES=['cerescopar', 'healthy', 'miner', 'phoma', 'rust']

rows = 5
columns = 5

for i in range(rows * columns):
    fig.add_subplot(rows, columns, i+1)
    num = random.randint(0, len(X)-1 )
    plt.imshow(X[num])
    plt.axis('off')
    plt.title(CLASSES[(np.argmax(y[num]))], fontsize=8)
plt.axis('off')
plt.show()
```


```
#   reshaping the images to 1D
X = X.reshape(-1, 128 * 128 * 3)

#   Oversampling method to remove imbalance class problem
X, y = SMOTETomek().fit_resample(X, y)

#   reshape images to images size of 208, 176, 3
X = X.reshape(-1, 128, 128, 3)

samples_after = len(X)
print("Number of samples after SMOTETomek :\t", sorted(Counter(np.argmax(y, axis=1)).items()))
```

```
Number of samples after SMOTETomek :     [(0, 2303), (1, 2303), (2, 2303), (3, 2303), (4, 2303)]
```

```
fig = plt.figure(figsize=(10,8))

rows = 5
columns = 5

for i in range(rows * columns):
    fig.add_subplot(rows, columns, i+1)
    num = random.randint(samples_before, samples_after - 1 )
    plt.imshow(X[num])
    plt.axis('off')
    plt.title(CLASSES[(np.argmax(y[num]))], fontsize=8)
plt.axis('off')
plt.show()
```




```
## sPLITTING


#   20% split to validation and 80% split to train set
X_train, x_val, y_train, y_val = train_test_split(X,y, test_size = 0.2)

#   20% split to test from 80% of train and 60% remains in train set
X_train, x_test, y_train, y_test = train_test_split(X_train,y_train, test_size = 0.2)

# Number of samples after train test split
print("Number of samples after splitting into Training, validation & test set\n")

print("Train     \t",sorted(Counter(np.argmax(y_train, axis=1)).items()))
print("Validation\t",sorted(Counter(np.argmax(y_val, axis=1)).items()))
print("Test      \t",sorted(Counter(np.argmax(y_test, axis=1)).items()))
```

```
Number of samples after splitting into Training, validation & test set

Train           [(0, 1494), (1, 1478), (2, 1442), (3, 1464), (4, 1491)]
Validation      [(0, 444), (1, 472), (2, 475), (3, 469), (4, 443)]
Test            [(0, 365), (1, 353), (2, 386), (3, 370), (4, 369)]
```

```
#   to free memeory we don't need this one as we split our data
del X, y


from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input


base_model_tf=ResNet50(include_top=False,weights='imagenet',input_shape=(128,128,3),classes=5)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernel
94765736/94765736 [==============================] - 1s 0us/step
```

```
#Model building
base_model_tf.trainable=False

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

```
model_resnet=Dense(5,activation='softmax')(model_resnet)


model_main=Model(inputs=pt,outputs=model_resnet)
model_main.summary()


### Model Compilation

# from tensorflow.keras.optimizers import SGD

model_main.compile(
    optimizer="Adam",
    loss = tf.keras.losses.CategoricalCrossentropy(name='loss'),
    # loss = "sparse_categorical_crossentropy",
    metrics=[
        tf.keras.metrics.CategoricalAccuracy(name='acc'),
        tf.keras.metrics.AUC(name='auc'),
        tfa.metrics.F1Score(num_classes=5),
        tf.metrics.Precision(name="precision"),
        tf.metrics.Recall(name="recall") ])
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 128, 128, 3)]     0

 tf.cast (TFOpLambda)        (None, 128, 128, 3)       0

 tf.__operators__.getitem (S  (None, 128, 128, 3)      0
 licingOpLambda)

 tf.nn.bias_add (TFOpLambda)  (None, 128, 128, 3)      0

 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 5)                 325

=================================================================
Total params: 23,858,565
Trainable params: 270,853
Non-trainable params: 23,587,712
_________________________________________________________________
```

```
from tensorflow import keras
model_main=keras.models.load_model('/content/drive/MyDrive/archive/model_f.h5')


#   declare to run on small gpu create batch sizes of images
valAug = ImageDataGenerator()

#   defining batch size
# batch_size = 8

hist = model_main.fit(valAug.flow(X_train, y_train, batch_size=64, shuffle = True),
steps_per_epoch=len(X_train)  // 80,
validation_data=valAug.flow(x_val, y_val, batch_size=64, shuffle = True),
validation_steps=len(x_test) //80,
epochs= 50,
batch_size=64,
# callbacks = CALLBACKS
)
```

```
57/57 [==============================] - 334s 6s/step - loss: 0.1729 - acc: 0.9451 - auc: 0.9953 - f1_score: 0.9447 - precision:
Epoch 31/50
57/57 [==============================] - 333s 6s/step - loss: 0.1747 - acc: 0.9404 - auc: 0.9953 - f1_score: 0.9398 - precision:
Epoch 32/50
57/57 [==============================] - 317s 6s/step - loss: 0.1617 - acc: 0.9433 - auc: 0.9959 - f1_score: 0.9425 - precision:
Epoch 33/50
57/57 [==============================] - 338s 6s/step - loss: 0.1938 - acc: 0.9354 - auc: 0.9943 - f1_score: 0.9351 - precision:
Epoch 34/50
57/57 [==============================] - 333s 6s/step - loss: 0.1820 - acc: 0.9382 - auc: 0.9947 - f1_score: 0.9377 - precision:
Epoch 35/50
57/57 [==============================] - 332s 6s/step - loss: 0.1950 - acc: 0.9293 - auc: 0.9944 - f1_score: 0.9282 - precision:
Epoch 36/50
57/57 [==============================] - 333s 6s/step - loss: 0.1938 - acc: 0.9290 - auc: 0.9941 - f1_score: 0.9281 - precision:
Epoch 37/50
57/57 [==============================] - 333s 6s/step - loss: 0.1692 - acc: 0.9418 - auc: 0.9955 - f1_score: 0.9411 - precision:
Epoch 38/50
57/57 [==============================] - 336s 6s/step - loss: 0.1697 - acc: 0.9468 - auc: 0.9954 - f1_score: 0.9463 - precision:
Epoch 39/50
57/57 [==============================] - 335s 6s/step - loss: 0.1609 - acc: 0.9451 - auc: 0.9958 - f1_score: 0.9450 - precision:
Epoch 40/50
57/57 [==============================] - 315s 6s/step - loss: 0.1700 - acc: 0.9423 - auc: 0.9953 - f1_score: 0.9417 - precision:
Epoch 41/50
57/57 [==============================] - 337s 6s/step - loss: 0.1828 - acc: 0.9354 - auc: 0.9950 - f1_score: 0.9349 - precision:
Epoch 42/50
57/57 [==============================] - 333s 6s/step - loss: 0.1616 - acc: 0.9465 - auc: 0.9959 - f1_score: 0.9461 - precision:
Epoch 43/50
57/57 [==============================] - 313s 6s/step - loss: 0.1777 - acc: 0.9393 - auc: 0.9953 - f1_score: 0.9392 - precision:
Epoch 44/50
57/57 [==============================] - 336s 6s/step - loss: 0.1914 - acc: 0.9379 - auc: 0.9941 - f1_score: 0.9374 - precision:
Epoch 45/50
57/57 [==============================] - 314s 6s/step - loss: 0.1625 - acc: 0.9420 - auc: 0.9957 - f1_score: 0.9415 - precision:
Epoch 46/50
57/57 [==============================] - 335s 6s/step - loss: 0.1578 - acc: 0.9484 - auc: 0.9961 - f1_score: 0.9477 - precision:
Epoch 47/50
57/57 [==============================] - 333s 6s/step - loss: 0.2018 - acc: 0.9282 - auc: 0.9934 - f1_score: 0.9276 - precision:
Epoch 48/50
57/57 [==============================] - 334s 6s/step - loss: 0.1851 - acc: 0.9331 - auc: 0.9946 - f1_score: 0.9328 - precision:
Epoch 49/50
57/57 [==============================] - 331s 6s/step - loss: 0.1559 - acc: 0.9445 - auc: 0.9962 - f1_score: 0.9437 - precision:
Epoch 50/50
57/57 [==============================] - 309s 5s/step - loss: 0.1530 - acc: 0.9456 - auc: 0.9961 - f1_score: 0.9455 - precision:
```

```
model_path= '/content/drive/MyDrive/plan/plant_coffee2.h5'
model_main.save(model_path)
```

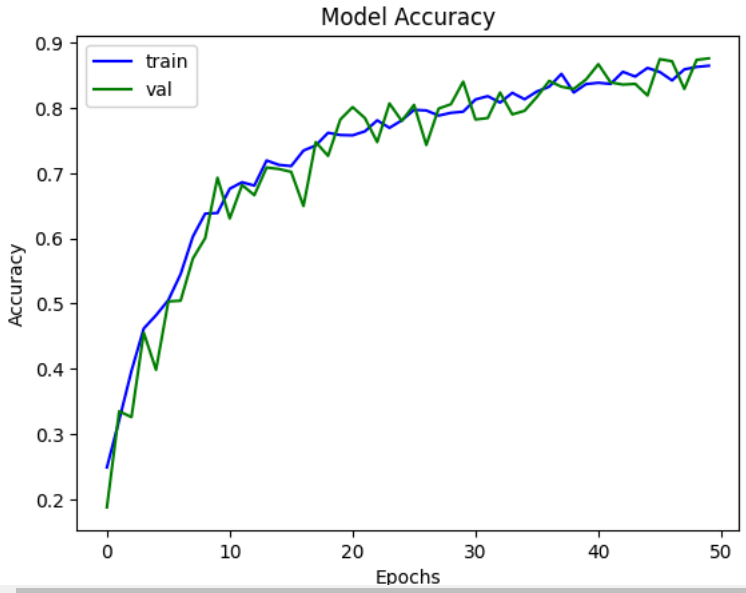
```
### Evaluate Model
test_scores = model_main.evaluate(x_test, y_test, batch_size = 32)
```

```
print("\n\nTesting Loss : \t\t {0:0.6f}".format(test_scores[0] ))
print("Testing Accuracy : \t {0:0.6f} %".format(test_scores[1] * 100))
print("Testing AC : \t\t {0:0.6f} %".format(test_scores[2] * 100))
print("Testing F1-Score : \t {0:0.6f} %".format(((test_scores[3][0] + test_scores[3][1] + test_scores[3][2] + test_scores[3][3] + test_s
print("Testing Precision : \t {0:0.6f} %".format(test_scores[4] * 100))
print("Testing Recall : \t {0:0.6f} %".format(test_scores[5] * 100))
```

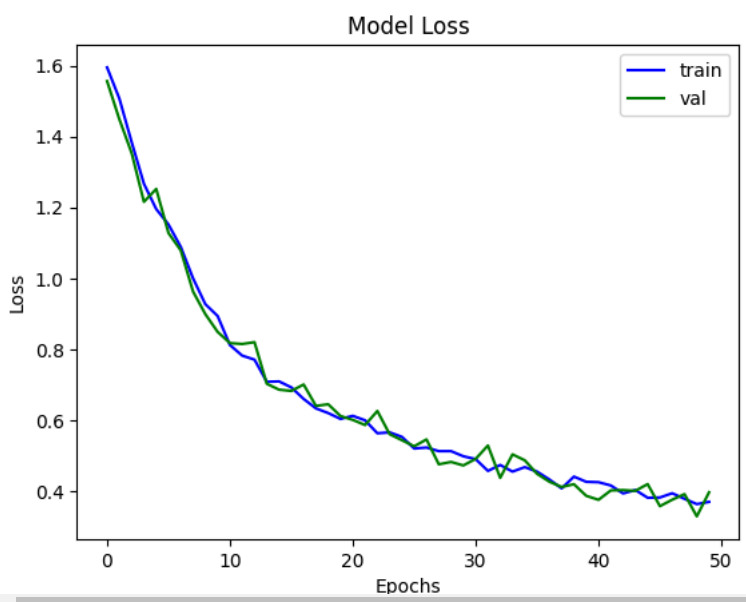
```
36/36 [==============================] - 76s 2s/step - loss: 0.1732 - acc: 0.9492 - auc: 0.9947 - f1_score: 0.9500 - precision: 0.95
```

```
Testing Loss :          0.173166
Testing Accuracy :      94.921190 %
Testing AC :            99.472862 %
Testing F1-Score :      95.004120 %
Testing Precision :     95.212764 %
Testing Recall :        94.045532 %
```

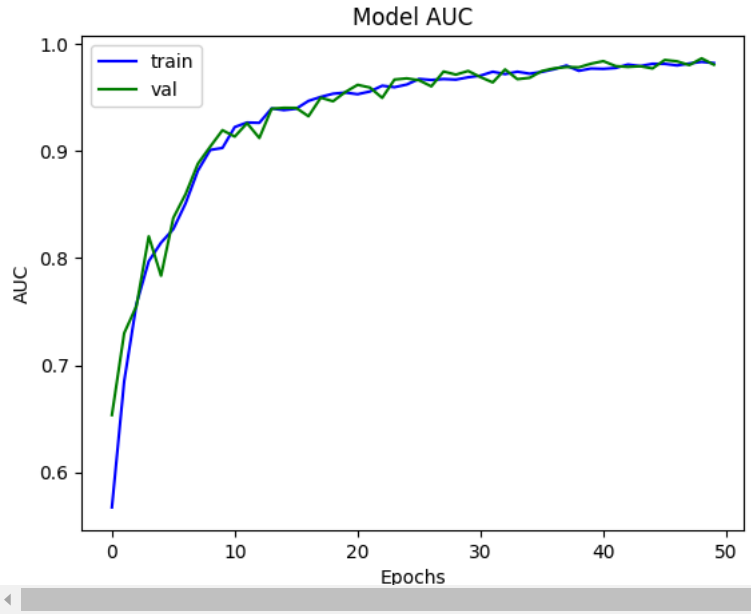
```
plt.plot(hist.history['acc'], 'b')
plt.plot(hist.history['val_acc'], 'g')
plt.title("Model Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend(["train", "val"])
plt.show()
```



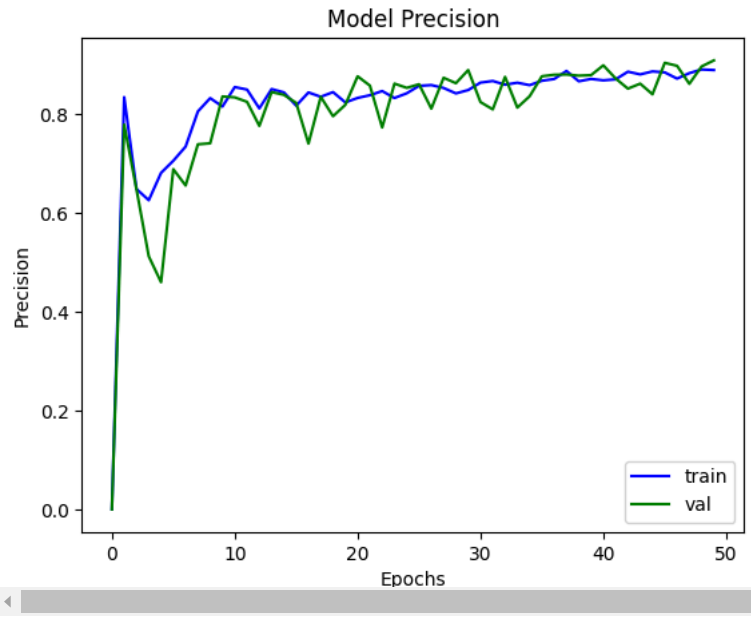

```
plt.plot(hist.history['loss'], 'b')
plt.plot(hist.history['val_loss'], 'g')
plt.title("Model Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend(["train", "val"])
plt.show()
```



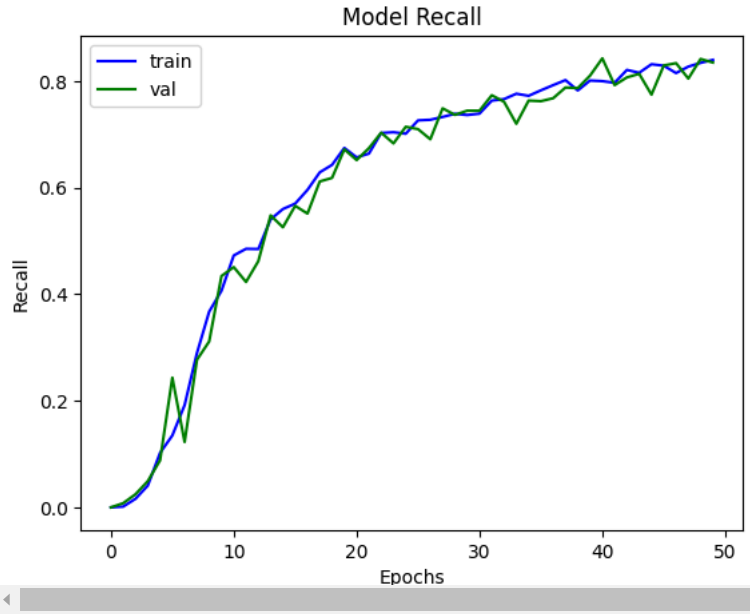

```
plt.plot(hist.history['auc'], 'b')
plt.plot(hist.history['val_auc'], 'g')
plt.title("Model AUC")
plt.xlabel("Epochs")
plt.ylabel("AUC")
plt.legend(["train", "val"])
plt.show()
```




```
plt.plot(hist.history['precision'], 'b')
plt.plot(hist.history['val_precision'], 'g')
plt.title("Model Precision")
plt.xlabel("Epochs")
plt.ylabel("Precision")
plt.legend(["train", "val"])
plt.show()
```



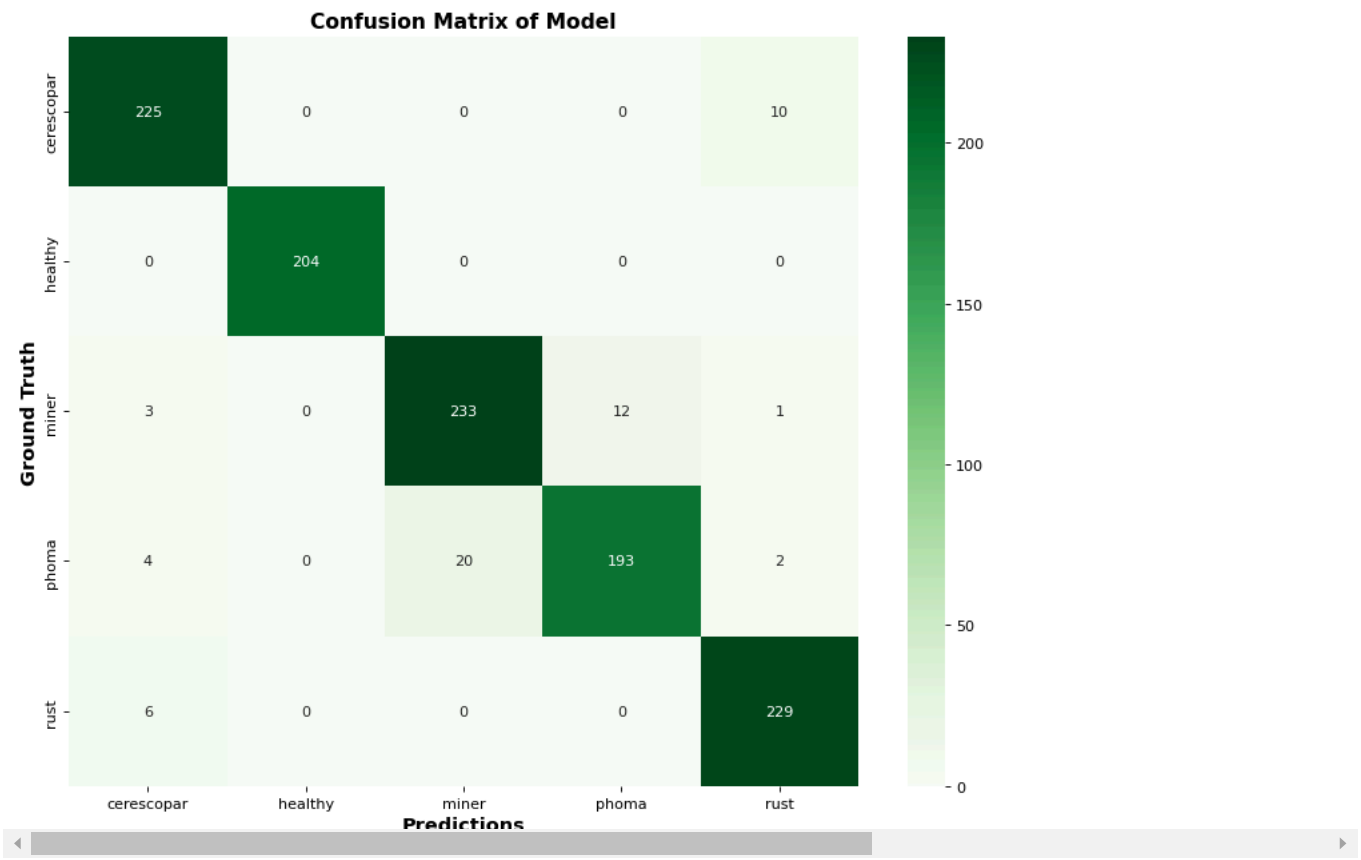

```
plt.plot(hist.history['recall'], 'b')
plt.plot(hist.history['val_recall'], 'g')
plt.title("Model Recall")
plt.xlabel("Epochs")
plt.ylabel("Recall")
plt.legend(["train", "val"])
plt.show()
```



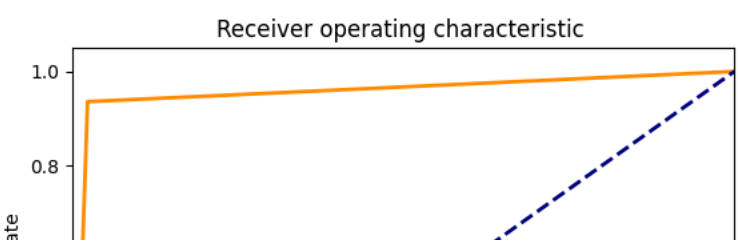

```
pred_labels = model_main.predict(x_test, batch_size=32)

def roundoff(arr):
    arr[np.argwhere(arr != arr.max())] = 0
    arr[np.argwhere(arr == arr.max())] = 1
    return arr

for labels in pred_labels:
    labels = roundoff(labels)

print(classification_report(y_test, pred_labels, target_names=CLASSES))
```

```
36/36 [==============================] - 75s 2s/step
                precision    recall  f1-score   support

       cerescopar       0.95      0.96      0.95       235
          healthy       1.00      1.00      1.00       204
            miner       0.92      0.94      0.93       249
            phoma       0.94      0.88      0.91       219
             rust       0.95      0.97      0.96       235

        micro avg       0.95      0.95      0.95      1142
        macro avg       0.95      0.95      0.95      1142
     weighted avg       0.95      0.95      0.95      1142
      samples avg       0.95      0.95      0.95      1142
```

```
pred_ls = np.argmax(pred_labels, axis=1)
test_ls = np.argmax(y_test, axis=1)

conf_arr = confusion_matrix(test_ls, pred_ls)

plt.figure(figsize=(10, 8), dpi=80, facecolor='w', edgecolor='k')

ax = sns.heatmap(conf_arr, cmap='Greens', annot=True, fmt='d', xticklabels= CLASSES, yticklabels=CLASSES)

plt.title('Confusion Matrix of Model', fontweight='bold', fontsize=14.0)
plt.xlabel('Predictions', fontweight='bold', fontsize=13)
plt.ylabel('Ground Truth', fontweight='bold', fontsize=13)
plt.tight_layout()
plt.show(ax)
```




```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(4):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], pred_labels[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), pred_labels.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

plt.figure()
lw = 2
plt.plot(
    fpr[2],
    tpr[2],
    color="darkorange",
    lw=lw,
    label="ROC curve (area = %0.4f)" % roc_auc[2])

plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic ")
plt.legend(loc="lower right")
plt.show()
```




```
from sklearn.metrics import accuracy_score

pred_labels = model_main.predict(x_test, batch_size=32)

def roundoff(arr):
    arr[np.argwhere(arr != arr.max())] = 0
    arr[np.argwhere(arr == arr.max())] = 1
    return arr

for labels in pred_labels:
    labels = roundoff(labels)

# Convert one-hot encoded predictions to integer labels
y_pred_labels = np.argmax(pred_labels, axis=1)

# Convert one-hot encoded ground truth labels to integer labels
y_true_labels = np.argmax(y_test, axis=1)

print(classification_report(y_true_labels, y_pred_labels, target_names=CLASSES))
accuracy = accuracy_score(y_true_labels, y_pred_labels)
print("Accuracy:", accuracy)
```

```
36/36 [==============================] - 76s 2s/step
                precision    recall  f1-score   support

       cerescopar       0.95      0.96      0.95       235
          healthy       1.00      1.00      1.00       204
            miner       0.92      0.94      0.93       249
            phoma       0.94      0.88      0.91       219
             rust       0.95      0.97      0.96       235

         accuracy                           0.95      1142
        macro avg       0.95      0.95      0.95      1142
     weighted avg       0.95      0.95      0.95      1142

     Accuracy: 0.9492119089316988
```

Start coding or generate with AI.

```
# Displaying error evolution during training (to see if the model learns or not)

acc_list = historyr.history['accuracy'] + historyr2.history['accuracy']
acc = np.array(acc_list)
plt.plot(acc, label='accuracy')
val_acc = historyr.history['val_accuracy'] + historyr2.history['val_accuracy']
plt.plot(val_acc, label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
```