```
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).
```

```
import os
import pandas as pd
import numpy as np
import random
import shutil
from shutil import copyfile
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import sklearn.model_selection
from matplotlib.offsetbox import (TextArea, DrawingArea, OffsetImage,AnnotationBbox)
import matplotlib.patches as mpatches
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
base_dir  = '/content/drive/MyDrive/Agricultural-crops'
os.chdir(base_dir)
```

```
directories_list = tf.io.gfile.listdir(base_dir)

# get number of labels
len_labels = len(directories_list)
print(f"Total Class Labels = {len_labels}")

vis_images = []; vis_labels =[]
length_file_list = []; label_list = []

for item in directories_list:

    # get each label directory
    item_dir = os.path.join(base_dir, item)
    # get list of images of each label
    item_files = os.listdir(item)
    len_per_label = len(os.listdir(item))

    length_file_list.append(len_per_label)
    label_list.append(item)

    # get first image of each label (for visualisation purpose)
    vis_images.append(os.path.join(item_dir, item_files[0]))
    # get respective label name (for visualisation purpose)
    vis_labels.append(item)

df_temp = pd.DataFrame({'Labels':label_list, 'Number of Images':length_file_list}).\
sort_values(by='Number of Images', ascending=False)
df_temp
```

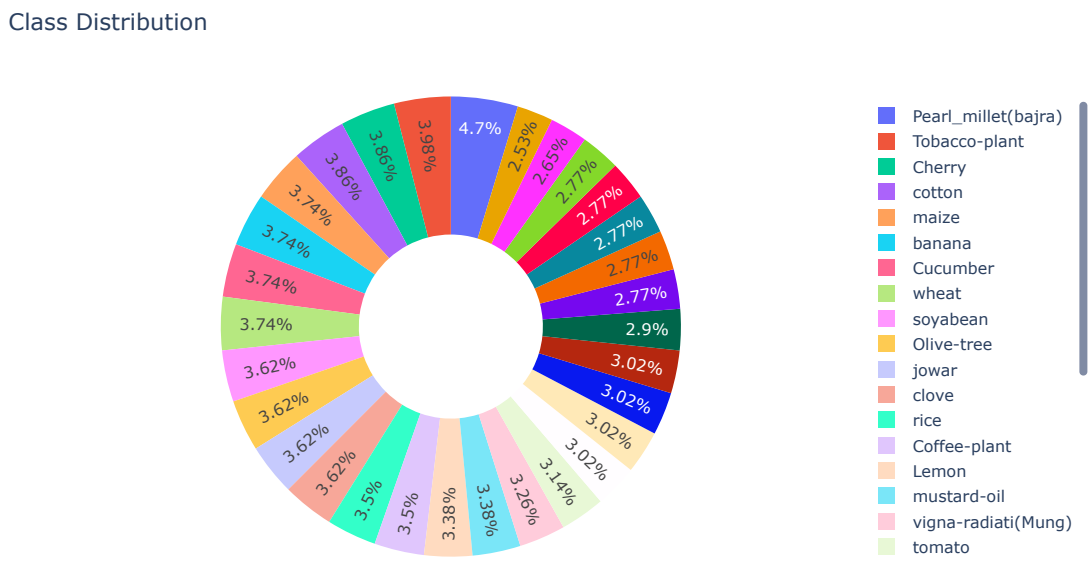
```
Total Class Labels = 30
```

	Labels	Number of Images
13	Pearl_millet(bajra)	39
2	Tobacco-plant	33
27	Cherry	32
20	cotton	32
15	maize	31
28	banana	31
29	Cucumber	31
4	wheat	31
3	soyabean	30
12	Olive-tree	30
14	jowar	30
23	clove	30
0	rice	29
22	Coffee-plant	29
11	Lemon	28

```
import plotly.express as px

class_names = df_temp['Labels']
class_dis = df_temp['Number of Images']

fig = px.pie(
    names=class_names,
    values=class_dis,
    title='Class Distribution',
    hole=0.4
)
fig.show()
```




```
plt.figure(figsize=(20,20))
for i in range(len(vis_labels)):
    plt.subplot(6,6,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    img = mpimg.imread(vis_images[i])
    plt.imshow(img)
    plt.xlabel(vis_labels[i])
    plt.suptitle(f"Classifying {len_labels} Types of Image Labels",fontsize=18, fontweight='bold')
plt.show()
```


rice


pineapple


Tobacco-plant


soyabean


sunflower


sugarcane


tomato


vigna-radiati(Mung)


Olive-tree


Pearl_millet(bajra)


jowar


maize


papaya


jute


cotton


coconut

almond


cardamom


chilli


Cherry

```
def split_data(SOURCE_DIR, TRAINING_DIR, VALIDATION_DIR, SPLIT_SIZE):

    selected_file_names = []
    all_file_names = os.listdir(SOURCE_DIR)
    for file_name in all_file_names:
        file_path = os.path.join(SOURCE_DIR, file_name)
        size = os.path.getsize(file_path)
        if size != 0:
              selected_file_names.append(file_name)
        else:
              print(f"{file_name} is zero length, so ignoring.")

    random.seed(42)
    selected_train_files = random.sample(selected_file_names, int(SPLIT_SIZE * len(selected_file_names)))
    selected_val_files = [x for x in selected_file_names if x not in selected_train_files]

    for file_name in selected_train_files:
        source = os.path.join(SOURCE_DIR, file_name)
        destination = os.path.join(TRAINING_DIR, file_name)
        copyfile(source, destination)

    for file_name in selected_val_files:
        source = os.path.join(SOURCE_DIR, file_name)
        destination = os.path.join(VALIDATION_DIR, file_name)
        copyfile(source, destination)


def create_train_val_dirs(root_path, split_size = 0.9):
    for item in directories_list:
        source_dir = os.path.join(base_dir, item)
        training_dir = os.path.join(root_path, f'_MODELLING/training/{item}')
        validation_dir = os.path.join(root_path, f'_MODELLING/validation/{item}')

        # Create EMPTY directory
        os.makedirs(training_dir)
        os.makedirs(validation_dir)

        split_data(source_dir, training_dir, validation_dir, split_size)
    print(f"Created training and validation directories containing images at split size of {split_size}")


create_train_val_dirs('/kaggle/working', split_size = 0.9)
```

```
 Created training and validation directories containing images at split size of 0.9
```

```
def show_ImageDataGenerator(vis_images, vis_labels, image_index):
    #Loads image in from the set image path
    class_label = vis_labels[image_index+5]
    img = tf.keras.preprocessing.image.load_img(vis_images[image_index+5], target_size= (250,250))
    img_tensor = tf.keras.preprocessing.image.img_to_array(img)
    img_tensor = np.expand_dims(img_tensor, axis=0)
    def show_image(datagen, param):
        pic = datagen.flow(img_tensor, batch_size =1)
        plt.figure(figsize=(10,3.5))
        #Plots our figures
        for i in range(1,4):
            plt.subplot(1, 3, i)
            batch = pic.next()
            image_ = batch[0].astype('uint8')
            plt.imshow(image_)
        plt.suptitle(f"Class: {class_label} \n Image Generator ({param})",fontsize=18, fontweight='bold')

        plt.show()
        datagen = ImageDataGenerator(rotation_range=30)
        show_image(datagen, "rotation_range=30")
        datagen = ImageDataGenerator(width_shift_range=0.2)
        show_image(datagen, "width_shift_range=0.2")

    datagen = ImageDataGenerator(zoom_range=0.2)
    show_image(datagen, "zoom_range=0.2")

    datagen = ImageDataGenerator(horizontal_flip=True)
    show_image(datagen, "horizontal_flip=True")

show_ImageDataGenerator(vis_images, vis_labels, image_index = 0)
```

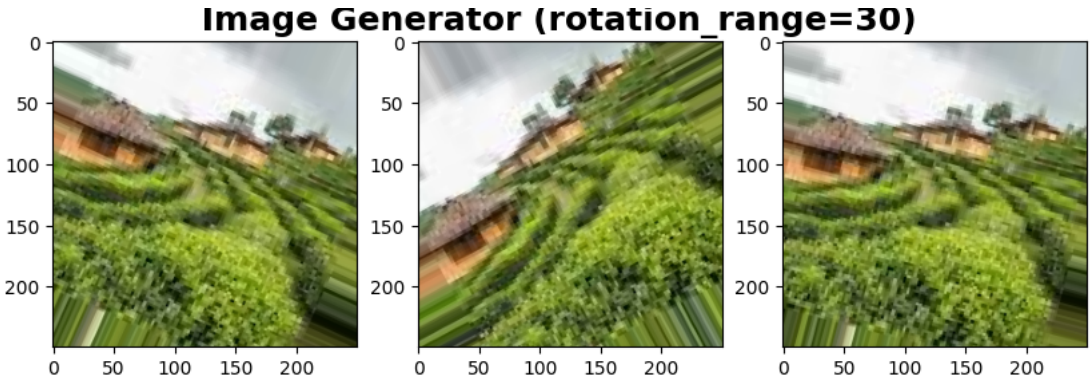
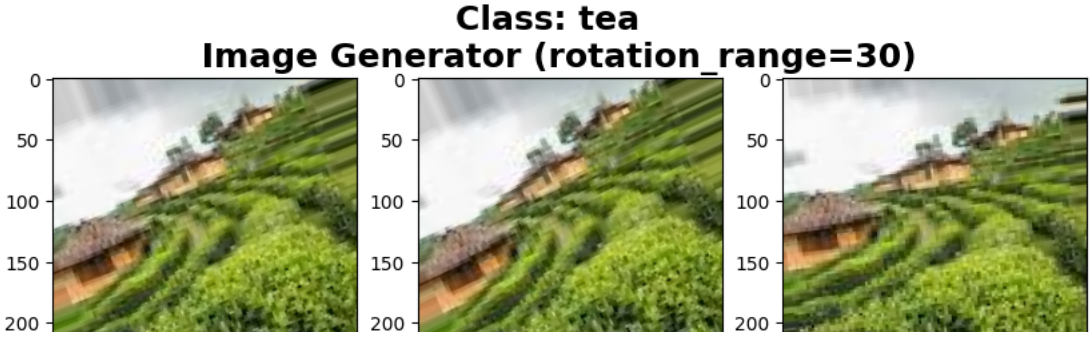
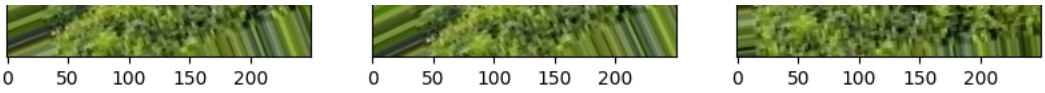
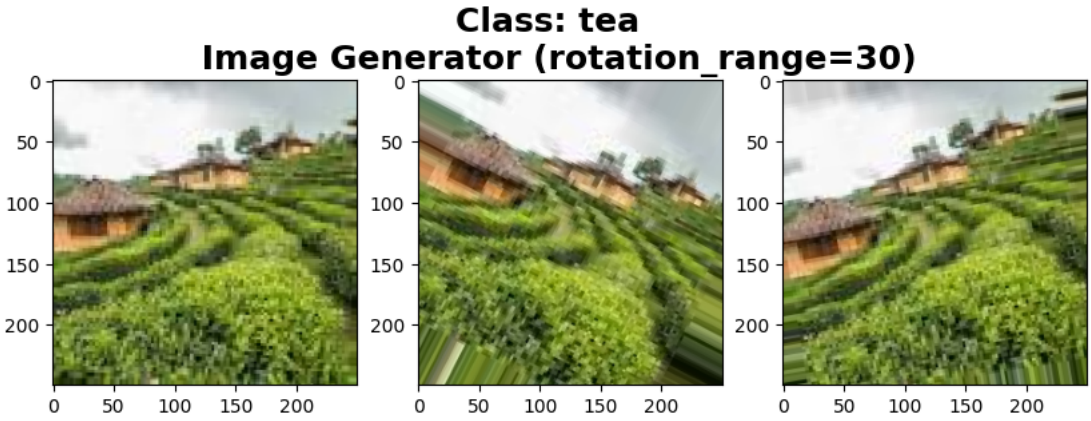
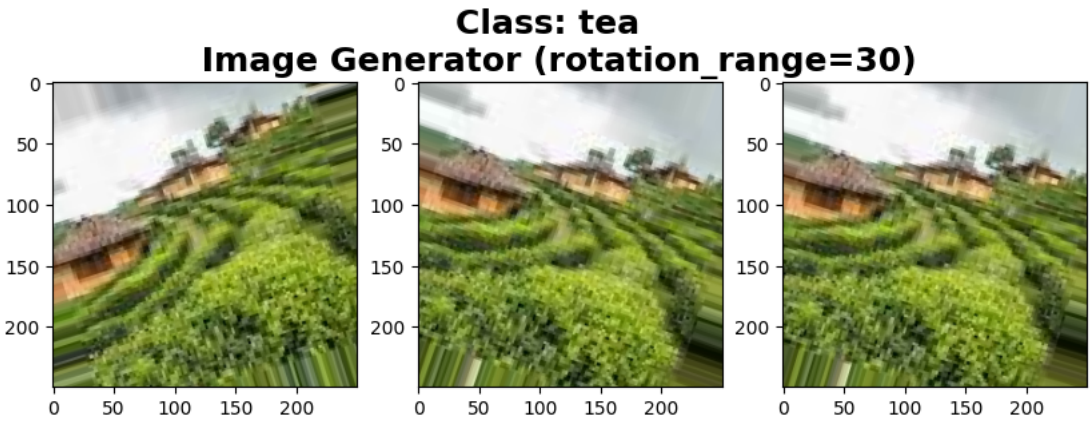


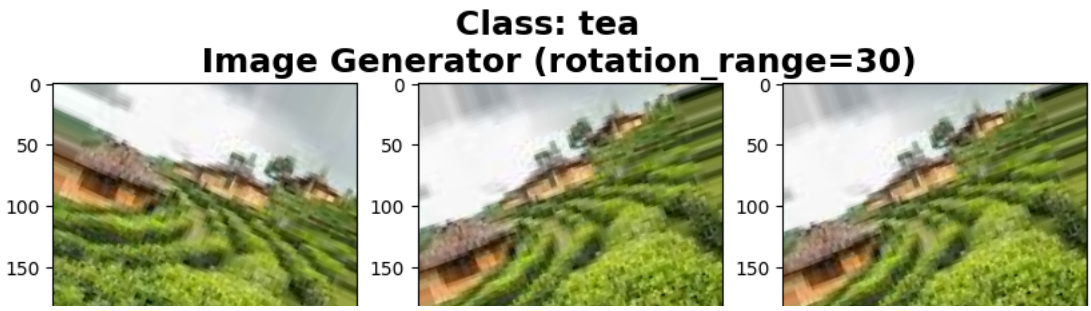
Image Generator (rotation_range=30)


Class: tea
Image Generator (rotation_range=30)




Class: tea
Image Generator (rotation_range=30)


Class: tea
Image Generator (rotation_range=30)


Class: tea
Image Generator (rotation_range=30)