```
# Set seeds for reproducibility
import random
random.seed(0)

import numpy as np
np.random.seed(0)

import tensorflow as tf
tf.random.set_seed(0)


import os
import json
from zipfile import ZipFile
from PIL import Image


!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.7)
```

```
kaggle_credentails = json.load(open("kaggle.json"))


# setup Kaggle API key as environment variables
os.environ['KAGGLE_USERNAME'] = kaggle_credentails["username"]
os.environ['KAGGLE_KEY'] = kaggle_credentails["key"]


!kaggle datasets download -d vipoooool/new-plant-diseases-dataset
```

```
Downloading new-plant-diseases-dataset.zip to /content
... resuming from 349175808 bytes (2548533379 bytes left) ...
100% 2.70G/2.70G [00:14<00:00, 199MB/s]
100% 2.70G/2.70G [00:14<00:00, 181MB/s]
```

```
!ls
```

```
 drive        'new plant diseases dataset(augmented)'   new-plant-diseases-dataset.zip
 kaggle.json  'New Plant Diseases Dataset(Augmented)'   test
```

```
# Unzip the downloaded dataset
with ZipFile("new-plant-diseases-dataset.zip", 'r') as zip_ref:
    zip_ref.extractall()


!ls
```

```
 drive        'new plant diseases dataset(augmented)'   new-plant-diseases-dataset.zip
 kaggle.json  'New Plant Diseases Dataset(Augmented)'   test
```

Importing Libraries

```
import tensorflow as tf
from tensorflow.keras.layers import Dense,Conv2D,MaxPool2D,Flatten,Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

DATA PREPROCESSING

Training Image

```
training_set = tf.keras.utils.image_dataset_from_directory(
    '/content/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 70295 files belonging to 38 classes.
```

VALIDATION IMAGE

```
validation_set = tf.keras.utils.image_dataset_from_directory(
    '/content/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 17572 files belonging to 38 classes.
```

BUILD MODEL

```
my_model = tf.keras.models.Sequential()
```

```
my_model.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_shape=[128,128,3]))
my_model.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,activation='relu'))
my_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

my_model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))
my_model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
my_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

my_model.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
my_model.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,activation='relu'))
my_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

my_model.add(tf.keras.layers.Conv2D(filters=256,kernel_size=3,padding='same',activation='relu'))
my_model.add(tf.keras.layers.Conv2D(filters=256,kernel_size=3,activation='relu'))
my_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

my_model.add(tf.keras.layers.Conv2D(filters=512,kernel_size=3,padding='same',activation='relu'))
my_model.add(tf.keras.layers.Conv2D(filters=512,kernel_size=3,activation='relu'))
my_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

```
my_model.add(tf.keras.layers.Dropout(0.25))                                          # avoid underfitting
```

```
my_model.add(tf.keras.layers.Flatten())
my_model.add(tf.keras.layers.Dense(units=1500,activation='relu'))
```

```
my_model.add(tf.keras.layers.Dropout(0.4))                                           #To avoid overfitting
```

```
#Output Layer
my_model.add(tf.keras.layers.Dense(units=38,activation='softmax'))
```

COMPLILING || TRAINING

```
my_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001) ,loss='categorical_crossentropy', metrics=['accuracy'])
```

```
my_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      896

 conv2d_1 (Conv2D)           (None, 126, 126, 32)      9248

 max_pooling2d (MaxPooling2  (None, 63, 63, 32)        0
 D)

 conv2d_2 (Conv2D)           (None, 63, 63, 64)        18496

 conv2d_3 (Conv2D)           (None, 61, 61, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 30, 30, 128)       73856

 conv2d_5 (Conv2D)           (None, 28, 28, 128)       147584

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 128)       0
 g2D)

 conv2d_6 (Conv2D)           (None, 14, 14, 256)       295168

 conv2d_7 (Conv2D)           (None, 12, 12, 256)       590080

 max_pooling2d_3 (MaxPoolin  (None, 6, 6, 256)         0
 g2D)

 conv2d_8 (Conv2D)           (None, 6, 6, 512)         1180160

 conv2d_9 (Conv2D)           (None, 4, 4, 512)         2359808

 max_pooling2d_4 (MaxPoolin  (None, 2, 2, 512)         0
 g2D)

 dropout (Dropout)           (None, 2, 2, 512)         0

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 1500)              3073500

 dropout_1 (Dropout)         (None, 1500)              0

 dense_1 (Dense)             (None, 38)                57038

=================================================================
Total params: 7842762 (29.92 MB)
Trainable params: 7842762 (29.92 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
training_history = my_model.fit(x=training_set,validation_data=validation_set,epochs=15)
```

```
Epoch 1/15
2197/2197 [==============================] - 178s 76ms/step - loss: 1.3585 - accuracy: 0.5993 - val_loss: 0.5001 - val_accuracy: 0.
Epoch 2/15
2197/2197 [==============================] - 162s 73ms/step - loss: 0.4533 - accuracy: 0.8557 - val_loss: 0.2891 - val_accuracy: 0.
Epoch 3/15
2197/2197 [==============================] - 161s 73ms/step - loss: 0.2623 - accuracy: 0.9160 - val_loss: 0.3160 - val_accuracy: 0.
Epoch 4/15
2197/2197 [==============================] - 162s 74ms/step - loss: 0.1837 - accuracy: 0.9406 - val_loss: 0.2627 - val_accuracy: 0.
Epoch 5/15
2197/2197 [==============================] - 160s 73ms/step - loss: 0.1384 - accuracy: 0.9534 - val_loss: 0.1387 - val_accuracy: 0.
Epoch 6/15
2197/2197 [==============================] - 161s 73ms/step - loss: 0.1081 - accuracy: 0.9644 - val_loss: 0.2453 - val_accuracy: 0.
Epoch 7/15
2197/2197 [==============================] - 158s 72ms/step - loss: 0.0843 - accuracy: 0.9725 - val_loss: 0.1351 - val_accuracy: 0.
Epoch 8/15
2197/2197 [==============================] - 160s 73ms/step - loss: 0.0752 - accuracy: 0.9750 - val_loss: 0.1149 - val_accuracy: 0.
Epoch 9/15
2197/2197 [==============================] - 161s 73ms/step - loss: 0.0640 - accuracy: 0.9793 - val_loss: 0.1353 - val_accuracy: 0.
Epoch 10/15
2197/2197 [==============================] - 162s 74ms/step - loss: 0.0573 - accuracy: 0.9816 - val_loss: 0.1115 - val_accuracy: 0.
Epoch 11/15
2197/2197 [==============================] - 162s 74ms/step - loss: 0.0533 - accuracy: 0.9829 - val_loss: 0.1271 - val_accuracy: 0.
```

```
Epoch 12/15
2197/2197 [==============================] - 163s 74ms/step - loss: 0.0437 - accuracy: 0.9860 - val_loss: 0.2143 - val_accuracy: 0.
Epoch 13/15
2197/2197 [==============================] - 162s 73ms/step - loss: 0.0416 - accuracy: 0.9870 - val_loss: 0.1156 - val_accuracy: 0.
Epoch 14/15
2197/2197 [==============================] - 160s 73ms/step - loss: 0.0382 - accuracy: 0.9879 - val_loss: 0.0960 - val_accuracy: 0.
Epoch 15/15
2197/2197 [==============================] - 161s 73ms/step - loss: 0.0371 - accuracy: 0.9881 - val_loss: 0.1088 - val_accuracy: 0.
```

EVALUATION OF THE MODEL

```
#Training set Accuracy
train_loss, train_acc = my_model.evaluate(training_set)
print('Training accuracy:', train_acc)
```

```
2197/2197 [==============================] - 55s 25ms/step - loss: 0.0213 - accuracy: 0.9941
Training accuracy: 0.9940536022186279
```

```
# Image Parameters
img_size = 224
batch_size = 32
```

```
#Validation set Accuracy
val_loss, val_acc = my_model.evaluate(validation_set)
print('Validation accuracy:', val_acc)
```

```
550/550 [==============================] - 14s 24ms/step - loss: 0.1088 - accuracy: 0.9711
Validation accuracy: 0.9710903763771057
```

VISUALISATION TOOLS

```
training_history.history
```


```
  0.2622852027416229,
  0.1837005615234375,
  0.13837198913097382,
  0.10814646631479263,
  0.08430405706167221,
  0.07515065371990204,
  0.0639662817120552,
  0.057262271642684937,
  0.05332052335143089,
  0.043691739439964294,
  0.041633713990449905,
  0.038166094571352005,
  0.03713954985141754],
 'accuracy': [0.5993313789367676,
  0.855665385723114,
  0.9159968495368958,
  0.9406074285507202,
  0.9534390568733215,
  0.9644213914871216,
  0.972544252872467,
  0.9750195741653442,
  0.9793015122413635,
  0.9816203117370605,
  0.9829148650169373,
  0.9860302805900574,
  0.9869834184646606,
  0.9879223108291626,
  0.9881499409675598],
 'val_loss': [0.5001130700111389,
  0.2891317903995514,
  0.31596770882606506,
  0.26269352436065674,
  0.13868436217308044,
  0.2452743500471115,
  0.13508851826190948,
  0.11488883942365646,
  0.13528084754943848,
  0.11148218810558319,
  0.12714561820030212,
  0.21434347331523895,
  0.1155521422624588,
  0.0960436537861824,
  0.10882437229156494],
 'val_accuracy': [0.8420214056968689,
  0.908490777015686,
  0.8964261412620544,
```

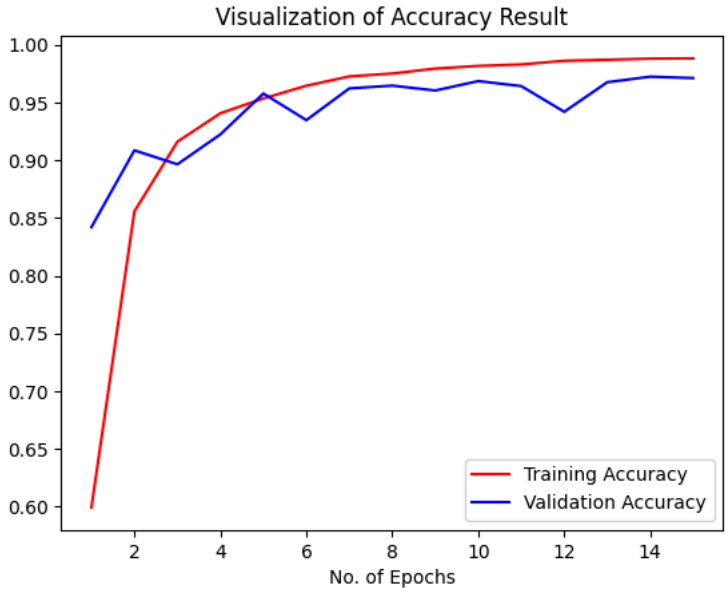
```
        0.9346687979379169,
        0.9621556997299194,
        0.9645458459854126,
        0.9603345990180969,
        0.9684725403785706,
        0.9641475081443787,
        0.9418962001800537,
        0.9675050973892212,
        0.9722854495048523,
        0.9710903763771057]}
```

```
import json
with open('training_hist.json','w') as f:
  json.dump(training_history.history,f)
```

```
print(training_history.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
epochs = [i for i in range(1,16)]
plt.plot(epochs,training_history.history['accuracy'],color='red',label='Training Accuracy')
plt.plot(epochs,training_history.history['val_accuracy'],color='blue',label='Validation Accuracy')
plt.xlabel('No. of Epochs')
plt.title('Visualization of Accuracy Result')
plt.legend()
plt.show()
```



Precision Recall Fscore

```
class_name = validation_set.class_names
print(class_name)
```

```
['Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust', 'Apple___healthy', 'Blueberry___healthy', 'Cherry_(includin
```

```
test_set = tf.keras.utils.image_dataset_from_directory(
    '/content/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=1,
    image_size=(128, 128),
    shuffle=False,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 17572 files belonging to 38 classes.
```

```
y_pred = my_model.predict(test_set)
predicted_categories = tf.argmax(y_pred, axis=1)
```

```
17572/17572 [==============================] - 55s 3ms/step
```

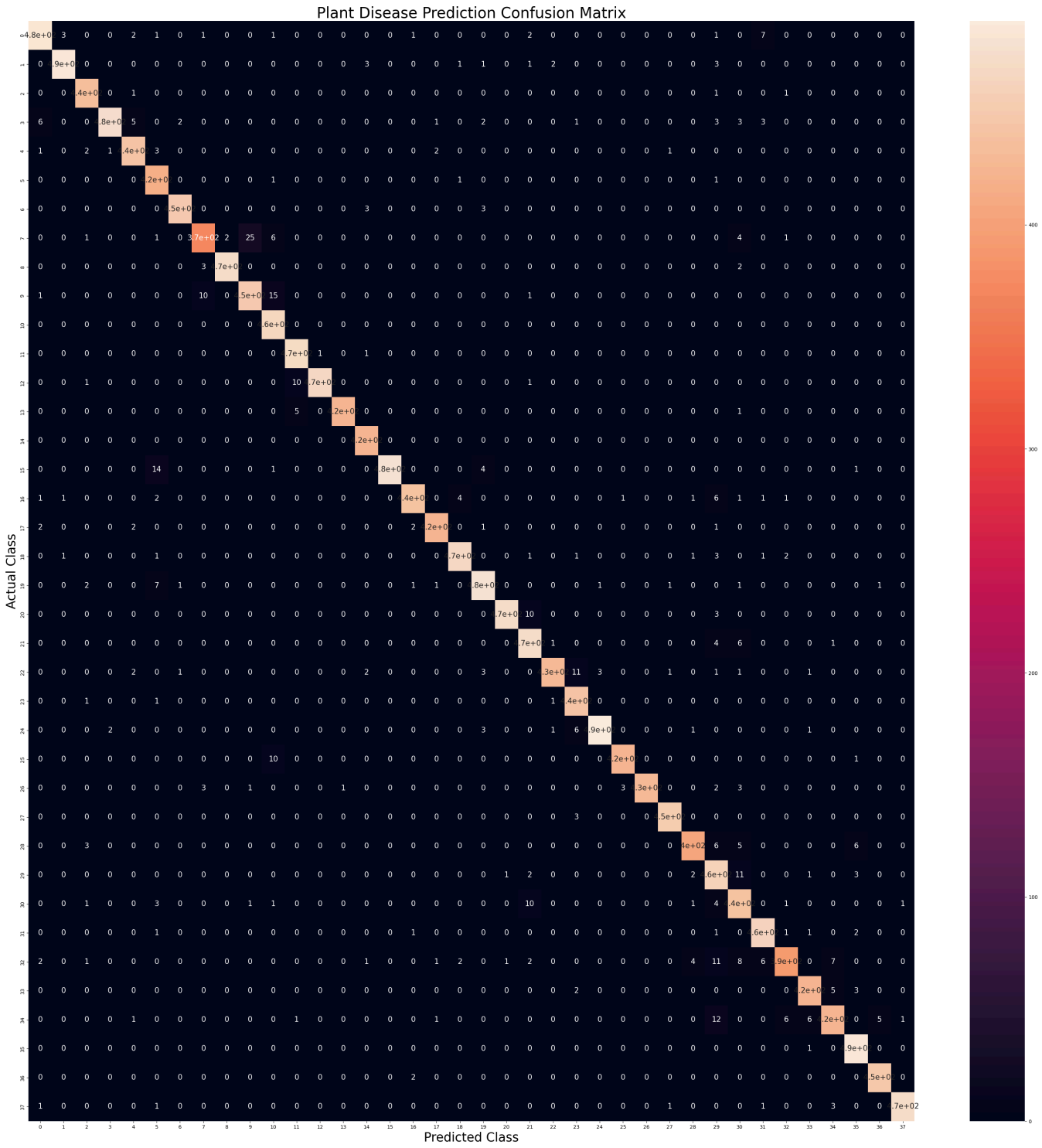
```
true_categories = tf.concat([y for x, y in test_set], axis=0)
Y_true = tf.argmax(true_categories, axis=1)
```

```
from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(Y_true,predicted_categories)

# Precision Recall Fscore
print(classification_report(Y_true,predicted_categories,target_names=class_name))
```

```
                                                 precision    recall  f1-score   support

                                  Apple___Apple_scab       0.97      0.96      0.97       504
                                   Apple___Black_rot       0.99      0.98      0.98       497
                            Apple___Cedar_apple_rust       0.97      0.99      0.98       440
                                     Apple___healthy       0.99      0.95      0.97       502
                                Blueberry___healthy       0.97      0.98      0.97       454
            Cherry_(including_sour)___Powdery_mildew       0.92      0.99      0.96       421
                   Cherry_(including_sour)___healthy       0.99      0.99      0.99       456
     Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.96      0.90      0.93       410
                         Corn_(maize)___Common_rust_       1.00      0.99      0.99       477
                Corn_(maize)___Northern_Leaf_Blight       0.94      0.94      0.94       477
                         Corn_(maize)___healthy       0.93      1.00      0.96       465
                              Grape___Black_rot       0.97      1.00      0.98       472
                   Grape___Esca_(Black_Measles)       1.00      0.97      0.99       480
          Grape___Leaf_blight_(Isariopsis_Leaf_Spot)       1.00      0.99      0.99       430
                               Grape___healthy       0.98      1.00      0.99       423
            Orange___Haunglongbing_(Citrus_greening)       1.00      0.96      0.98       503
                         Peach___Bacterial_spot       0.98      0.96      0.97       459
                                Peach___healthy       0.99      0.98      0.98       432
                  Pepper,_bell___Bacterial_spot       0.98      0.98      0.98       478
                         Pepper,_bell___healthy       0.97      0.97      0.97       497
                          Potato___Early_blight       1.00      0.97      0.98       485
                           Potato___Late_blight       0.94      0.98      0.96       485
                               Potato___healthy       0.99      0.94      0.97       456
                            Raspberry___healthy       0.95      0.99      0.97       445
                              Soybean___healthy       0.99      0.97      0.98       505
                        Squash___Powdery_mildew       0.99      0.97      0.98       434
                       Strawberry___Leaf_scorch       1.00      0.97      0.99       444
                           Strawberry___healthy       0.99      0.99      0.99       456
                         Tomato___Bacterial_spot       0.98      0.95      0.96       425
                           Tomato___Early_blight       0.88      0.96      0.92       480
                            Tomato___Late_blight       0.91      0.95      0.93       463
                              Tomato___Leaf_Mold       0.96      0.99      0.97       470
                     Tomato___Septoria_leaf_spot       0.97      0.89      0.93       436
   Tomato___Spider_mites Two-spotted_spider_mite       0.97      0.98      0.98       435
                           Tomato___Target_Spot       0.96      0.93      0.95       457
          Tomato___Tomato_Yellow_Leaf_Curl_Virus       0.97      1.00      0.98       490
                    Tomato___Tomato_mosaic_virus       0.99      1.00      0.99       448
                                Tomato___healthy       1.00      0.99      0.99       481

                                        accuracy                           0.97     17572
                                       macro avg       0.97      0.97      0.97     17572
                                    weighted avg       0.97      0.97      0.97     17572
```

Confusion Matrix Visualise

```
plt.figure(figsize=(40, 40))
sns.heatmap(cm,annot=True,annot_kws={"size": 15})

plt.xlabel('Predicted Class',fontsize = 25)
plt.ylabel('Actual Class',fontsize = 25)
plt.title('Plant Disease Prediction Confusion Matrix',fontsize = 30)
plt.show()
```



SAVE MODEL

```
my_model.save('/content/drive/MyDrive/trained_plant_disease.keras')
```