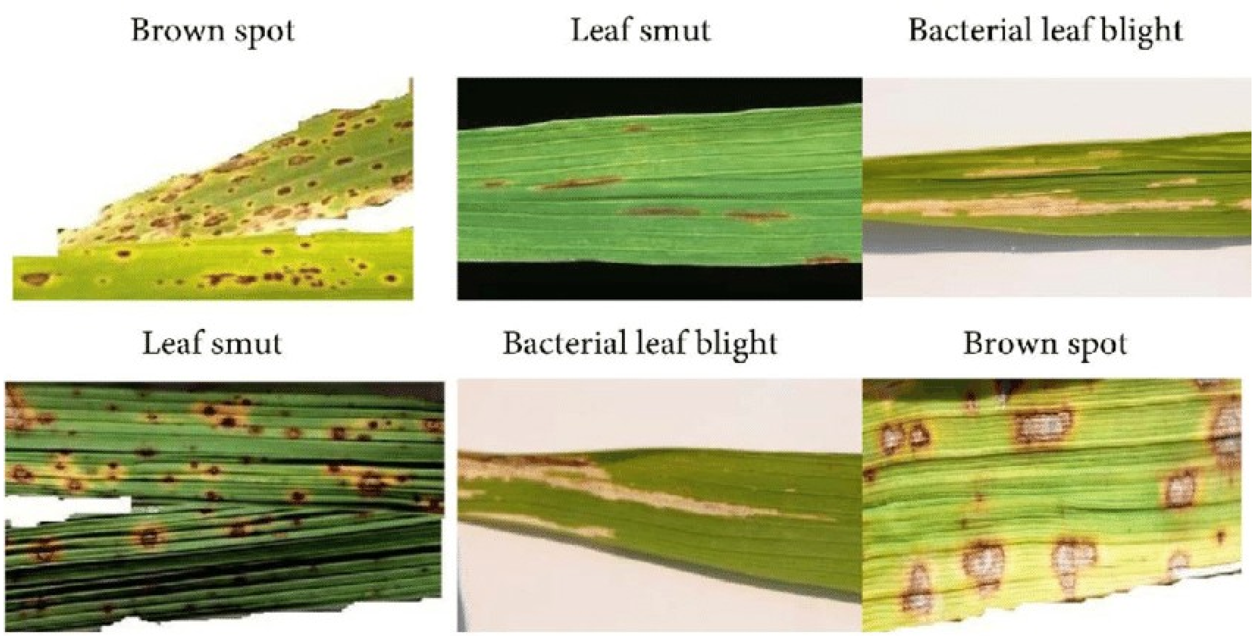
DATA SCIENCE PROJECT ON RICE LEAF DISEASE DETECTION

INTRODUCTION

The rice leaf suffers from several bacterial, viral, or fungal diseases and these diseases reduce rice production significantly. To sustain rice demand for a vast population globally.The rice leaves related diseases often pose threats to the sustainable production of rice affecting many farmers around the world. Early diagnosis and appropriate remedy of the rice leaf infection is crucial in facilitating healthy growth of the rice plants to ensure adequate supply and food security to the rapidly increasing population.

RICE LEAF DISEASE:



WE HAVE DEVICE THE PROJECT INTO MULTIPLE STEPS

- Importing library
- Loading data
- Preparing data
- Data Processing
- Model building
- Training
- Evaluation
- Testing

DATA SUMMARY

This dataset contains 120 jpg images of disease infected rice leaves. The images are grouped into 3 classes based on the type of disease. There are 40 images in each class.

Classes

- Leaf smut
- Brown spot
- Bacterial leaf blight

PYTHON IMPLIMENTATION

IMPORTING NECESSARY LIBRARY

```
import numpy as np
import keras
from tensorflow import keras
import tensorflow as tf
from tensorflow.keras import layers
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
import matplotlib.pyplot as plt
%matplotlib inline
import random
import cv2
import os
from PIL import Image
import warnings
warnings.filterwarnings('ignore')

## To connect Google Drive (GDrive) with Colab
# Step:2 Mount drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

MAKE SUBSET OF TRAIN, TEST, VALIDATION

```
import splitfolders
splitfolders.ratio(r"C:\Users\hp\Downloads\PRCP-1001-RiceLeaf\Data\
bacterical disease", output="output", seed=1337, ratio=(.8, 0.1,0.1))
```

```
Copying files: 120 files [00:00, 749.04 files/s]
```

```
# Sorting the path of data into veriable
train_dir =
'/content/drive/MyDrive/PRCP-1001-RiceLeaf/Data/output/train'
#Location of training images
validation_dir =
```

```
'/content/drive/MyDrive/PRCP-1001-RiceLeaf/Data/output/val' # Location of test images
test_dir ='/content/drive/MyDrive/PRCP-1001-RiceLeaf/Data/output/test' # Location of test images
```

GENERATING TRAINING AND VALIDATION BATCHES OF IMAGES

```
# Generating batches of image data
train_datagen = ImageDataGenerator(
    rescale= (1./255),
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

val_datagen = ImageDataGenerator(rescale=(1./255))




train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(180,180),
    batch_size=16,
    color_mode='rgb',
    class_mode='categorical')


val_generator = val_datagen.flow_from_directory(
    validation_dir,
    target_size=(180,180),
    batch_size=16,
    color_mode='rgb',
    class_mode='categorical')
```

```
Found 96 images belonging to 3 classes.
Found 12 images belonging to 3 classes.
```

PLOTTING TRAIN IMAGES WITH THEIR LABELS

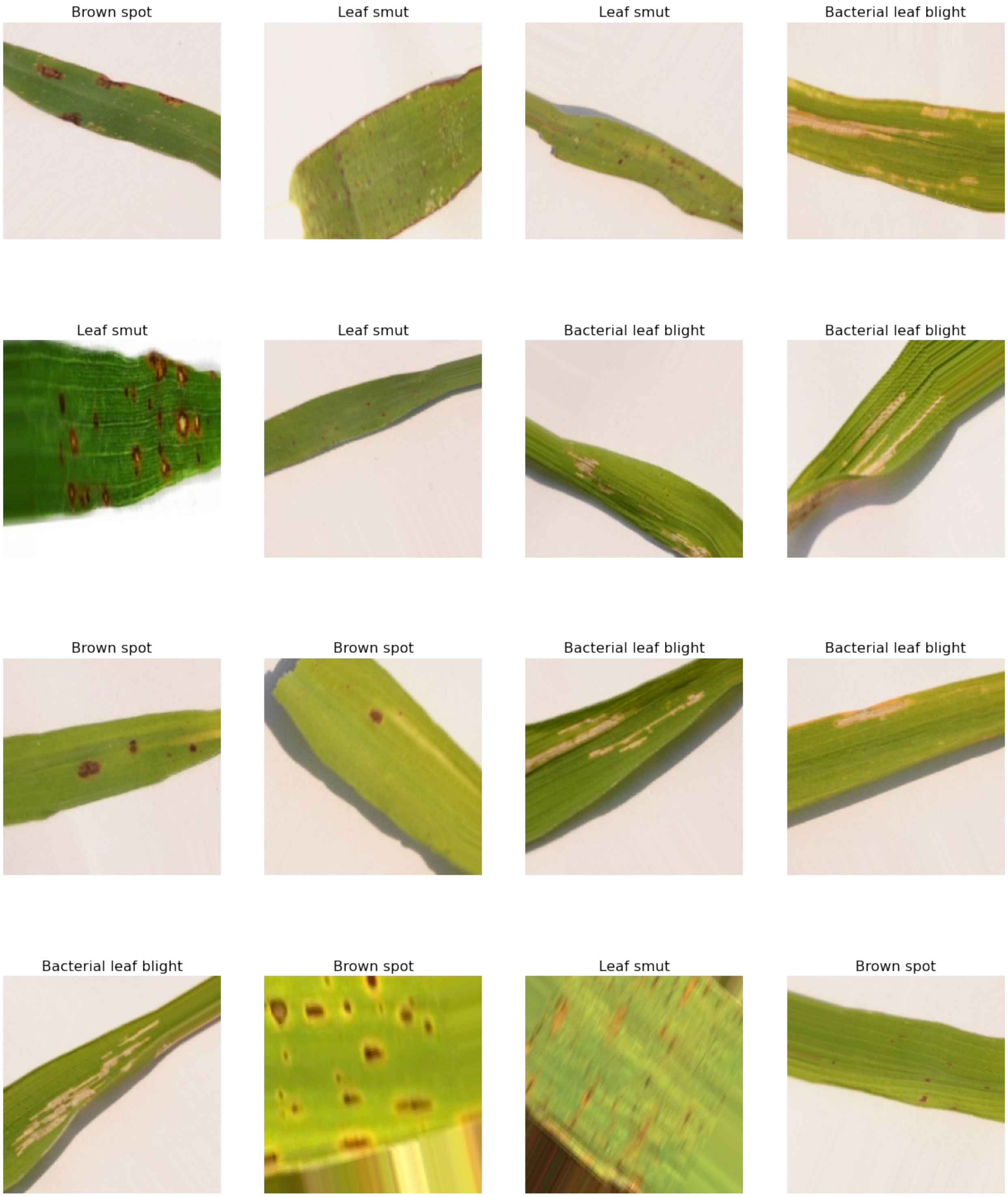
```
# plotting train images with their labels
def plots(ims, figsize=(20,25), rows=4, interp = False, title = None):
```

```
    f = plt.figure(figsize=figsize)
    cols = len(ims) // rows if len(ims) % 2 ==0 else len(ims) //
rows+1
    for i in range(len(ims)):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('off')

sp.set_title(class_names[title[i].tolist().index(1)],fontsize=16)
        plt.imshow(ims[i])


# Make list of classes
class_names = ['Bacterial leaf blight', 'Brown spot', 'Leaf smut']

imgs, labels = next(train_generator)
plots(imgs, title = labels)
```



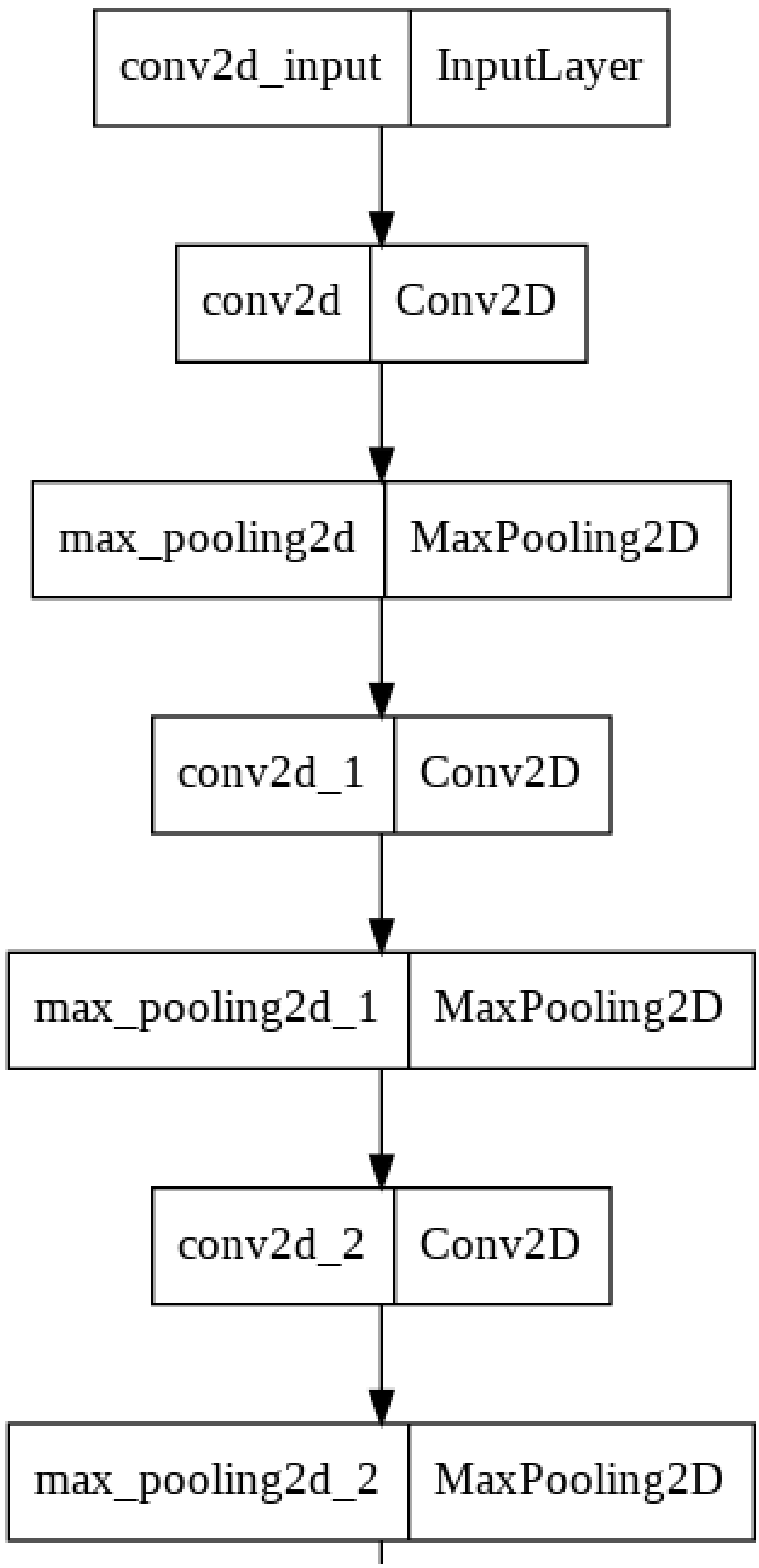
CNN MODEL ARCHITECTURE

```
from keras.layers.core.activation import Activation
from keras import models, layers
model = models.Sequential()
model.add(layers.Conv2D(filters=32, kernel_size=(3,3),
```

```
activation='relu',input_shape=(180,180,3)))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(filters=64,kernel_size=(3,3),activation=
'relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(filters=128,kernel_size=(3,3),activation=
'relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(filters=256,kernel_size=(3,3),activation=
'relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Dropout(rate=0.5))
model.add(layers.Flatten())
model.add(layers.Dense(3, activation ='softmax'))
```

PLOTTING GRAPHICAL REPRESENTATION OF MODEL

```
import keras
import pydotplus
from keras.utils.vis_utils import model_to_dot
keras.utils.vis_utils.plot_model(model)
```


conv2d_input
InputLayer
conv2d
Conv2D
max_pooling2d
MaxPooling2D
conv2d_1
Conv2D
max_pooling2d_1
MaxPooling2D
conv2d_2
Conv2D
max_pooling2d_2
MaxPooling2D

SUMMARY OF MODEL

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 178, 178, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 89, 89, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 87, 87, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 43, 43, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 41, 41, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 20, 20, 128)      0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 18, 18, 256)       295168    
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 9, 9, 256)        0         
 2D)                                                             
                                                                 
 dropout (Dropout)           (None, 9, 9, 256)         0         
                                                                 
 flatten (Flatten)           (None, 20736)             0         
                                                                 
 dense (Dense)               (None, 3)                 62211     
                                                                 
=================================================================
Total params: 450,627
Trainable params: 450,627
Non-trainable params: 0
_________________________________________________________________
```

COMPILE MODEL

```
from tensorflow.keras import optimizers
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

TRAIN MODEL

```
# Fitting the data

history = model.fit_generator(train_generator,
    epochs=30, # epochs used to how many itertion (1fp + loss + 1bp)
    validation_data = val_generator,
)
```

```
Epoch 1/30
6/6 [==============================] - 4s 669ms/step - loss: 0.6364 -
accuracy: 0.6667 - val_loss: 0.4493 - val_accuracy: 0.8333
Epoch 2/30
6/6 [==============================] - 4s 618ms/step - loss: 0.5945 -
accuracy: 0.6875 - val_loss: 0.3712 - val_accuracy: 0.9167
Epoch 3/30
6/6 [==============================] - 4s 601ms/step - loss: 0.5315 -
accuracy: 0.7812 - val_loss: 0.3293 - val_accuracy: 1.0000
Epoch 4/30
6/6 [==============================] - 4s 614ms/step - loss: 0.5108 -
accuracy: 0.7604 - val_loss: 0.5318 - val_accuracy: 0.6667
Epoch 5/30
6/6 [==============================] - 4s 621ms/step - loss: 0.7039 -
accuracy: 0.7500 - val_loss: 0.5219 - val_accuracy: 0.7500
Epoch 6/30
6/6 [==============================] - 4s 622ms/step - loss: 0.6213 -
accuracy: 0.7292 - val_loss: 1.1733 - val_accuracy: 0.6667
Epoch 7/30
6/6 [==============================] - 4s 614ms/step - loss: 0.5475 -
accuracy: 0.7500 - val_loss: 0.3977 - val_accuracy: 0.7500
Epoch 8/30
6/6 [==============================] - 4s 624ms/step - loss: 0.5748 -
accuracy: 0.7188 - val_loss: 0.3949 - val_accuracy: 0.9167
Epoch 9/30
6/6 [==============================] - 4s 618ms/step - loss: 0.5453 -
accuracy: 0.7500 - val_loss: 0.6492 - val_accuracy: 0.5833
Epoch 10/30
6/6 [==============================] - 4s 627ms/step - loss: 0.5564 -
accuracy: 0.7812 - val_loss: 0.8423 - val_accuracy: 0.8333
Epoch 11/30
6/6 [==============================] - 4s 617ms/step - loss: 0.4903 -
accuracy: 0.7812 - val_loss: 0.8719 - val_accuracy: 0.6667
Epoch 12/30
6/6 [==============================] - 4s 597ms/step - loss: 0.4593 -
accuracy: 0.7604 - val_loss: 1.3835 - val_accuracy: 0.7500
Epoch 13/30
6/6 [==============================] - 4s 603ms/step - loss: 0.3947 -
accuracy: 0.8021 - val_loss: 0.7953 - val_accuracy: 0.8333
Epoch 14/30
6/6 [==============================] - 4s 611ms/step - loss: 0.3515 -
```

```
accuracy: 0.8333 - val_loss: 0.3332 - val_accuracy: 0.8333
Epoch 15/30
6/6 [==============================] - 4s 597ms/step - loss: 0.3721 -
accuracy: 0.8125 - val_loss: 0.7977 - val_accuracy: 0.8333
Epoch 16/30
6/6 [==============================] - 4s 621ms/step - loss: 0.4631 -
accuracy: 0.8021 - val_loss: 0.7034 - val_accuracy: 0.8333
Epoch 17/30
6/6 [==============================] - 4s 634ms/step - loss: 0.3942 -
accuracy: 0.8125 - val_loss: 0.2659 - val_accuracy: 0.9167
Epoch 18/30
6/6 [==============================] - 4s 623ms/step - loss: 0.3746 -
accuracy: 0.8229 - val_loss: 0.2807 - val_accuracy: 0.8333
Epoch 19/30
6/6 [==============================] - 4s 620ms/step - loss: 0.3677 -
accuracy: 0.8438 - val_loss: 0.6246 - val_accuracy: 0.7500
Epoch 20/30
6/6 [==============================] - 4s 617ms/step - loss: 0.4460 -
accuracy: 0.7812 - val_loss: 0.2445 - val_accuracy: 0.9167
Epoch 21/30
6/6 [==============================] - 4s 623ms/step - loss: 0.3715 -
accuracy: 0.8333 - val_loss: 0.2997 - val_accuracy: 0.8333
Epoch 22/30
6/6 [==============================] - 4s 616ms/step - loss: 0.3389 -
accuracy: 0.8646 - val_loss: 0.2974 - val_accuracy: 0.8333
Epoch 23/30
6/6 [==============================] - 4s 614ms/step - loss: 0.3231 -
accuracy: 0.8542 - val_loss: 1.9464 - val_accuracy: 0.5833
Epoch 24/30
6/6 [==============================] - 4s 620ms/step - loss: 0.3299 -
accuracy: 0.8646 - val_loss: 0.1750 - val_accuracy: 0.9167
Epoch 25/30
6/6 [==============================] - 4s 603ms/step - loss: 0.3549 -
accuracy: 0.8646 - val_loss: 0.3840 - val_accuracy: 0.8333
Epoch 26/30
6/6 [==============================] - 4s 622ms/step - loss: 0.4918 -
accuracy: 0.7917 - val_loss: 0.4623 - val_accuracy: 0.8333
Epoch 27/30
6/6 [==============================] - 4s 606ms/step - loss: 0.4032 -
accuracy: 0.8333 - val_loss: 0.4246 - val_accuracy: 0.7500
Epoch 28/30
6/6 [==============================] - 4s 630ms/step - loss: 0.4361 -
accuracy: 0.8750 - val_loss: 2.5044 - val_accuracy: 0.8333
Epoch 29/30
6/6 [==============================] - 4s 621ms/step - loss: 0.3579 -
accuracy: 0.8750 - val_loss: 0.3632 - val_accuracy: 0.8333
Epoch 30/30
6/6 [==============================] - 4s 615ms/step - loss: 0.3510 -
accuracy: 0.8438 - val_loss: 0.2692 - val_accuracy: 0.9167
```

AFTER TRAINING

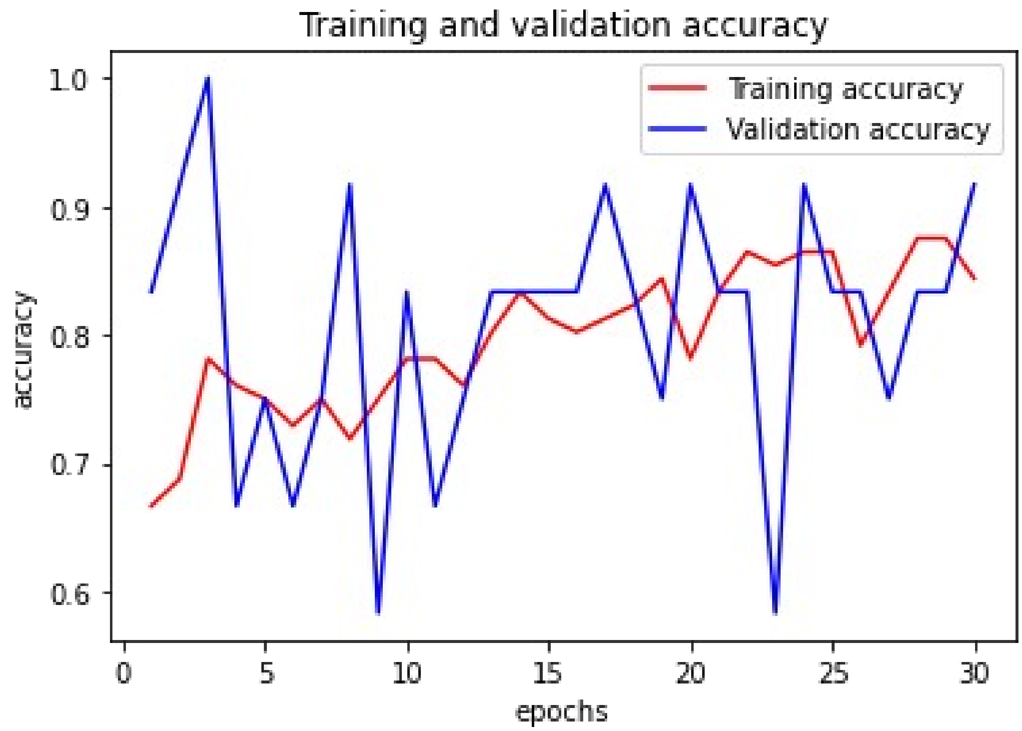
- Validation accuracy.91.67%
- Training accuracy.84.38%

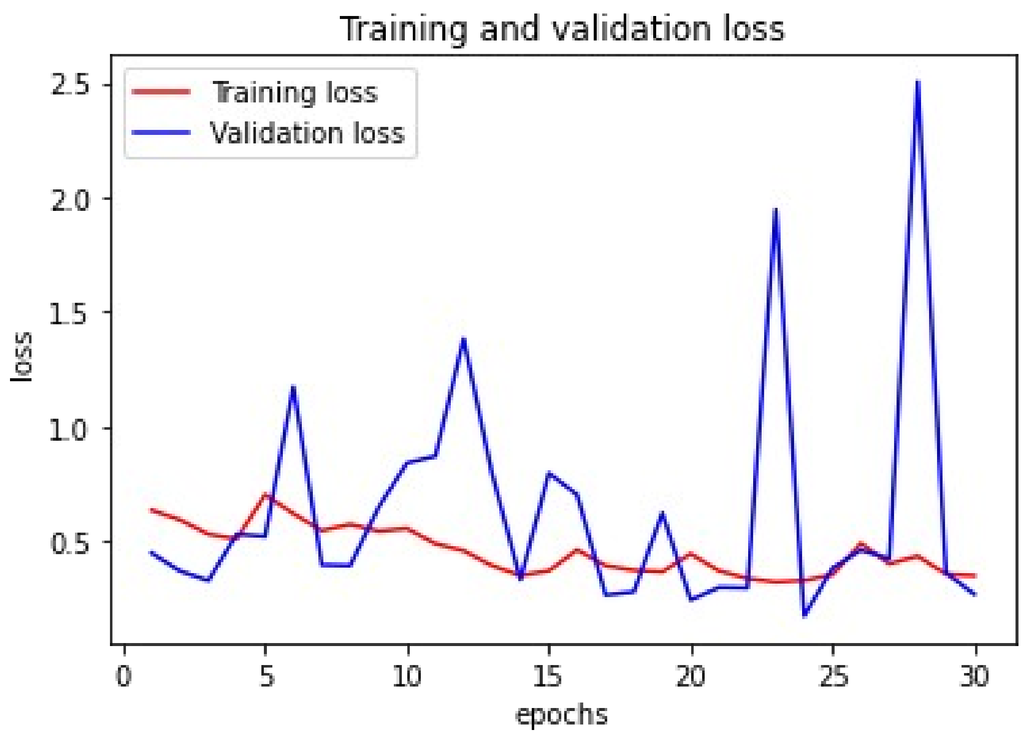
MODEL SAVING

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

PLOTTING THE TRAINING ACCURACY AND VALIDATION ACCURACY AND TRAINING LOSS AND VALIDATION LOSS

```
# Step:9 Plotting the training accuracy and validation accuracy
# Plotting the traning loss and validation loss
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "r", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "r", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend()
plt.show()
```





CREATED MODEL SUMMARY

```
model = tf.keras.models.load_model("model.h5")
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 178, 178, 32)      896       
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 89, 89, 32)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 87, 87, 64)        18496     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 43, 43, 64)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 41, 41, 128)       73856     
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 20, 20, 128)      0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 18, 18, 256)       295168    
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 9, 9, 256)        0         
 2D)                                                             
                                                                 
 dropout_2 (Dropout)         (None, 9, 9, 256)         0         
                                                                 
 flatten_1 (Flatten)         (None, 20736)             0         
                                                                 
 dense_2 (Dense)             (None, 256)               5308672   
                                                                 
 dropout_3 (Dropout)         (None, 256)               0         
                                                                 
 dense_3 (Dense)             (None, 3)                 771       
                                                                 
=================================================================
Total params: 5,697,859
Trainable params: 5,697,859
Non-trainable params: 0
_________________________________________________________________
```

EVALUATION AND TESTING MODEL

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


test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(180,180),
    batch_size=16,
```

```
    color_mode='rgb',
    class_mode='categorical')
```

```
Found 12 images belonging to 3 classes.
```

```
model.evaluate(test_generator)
```

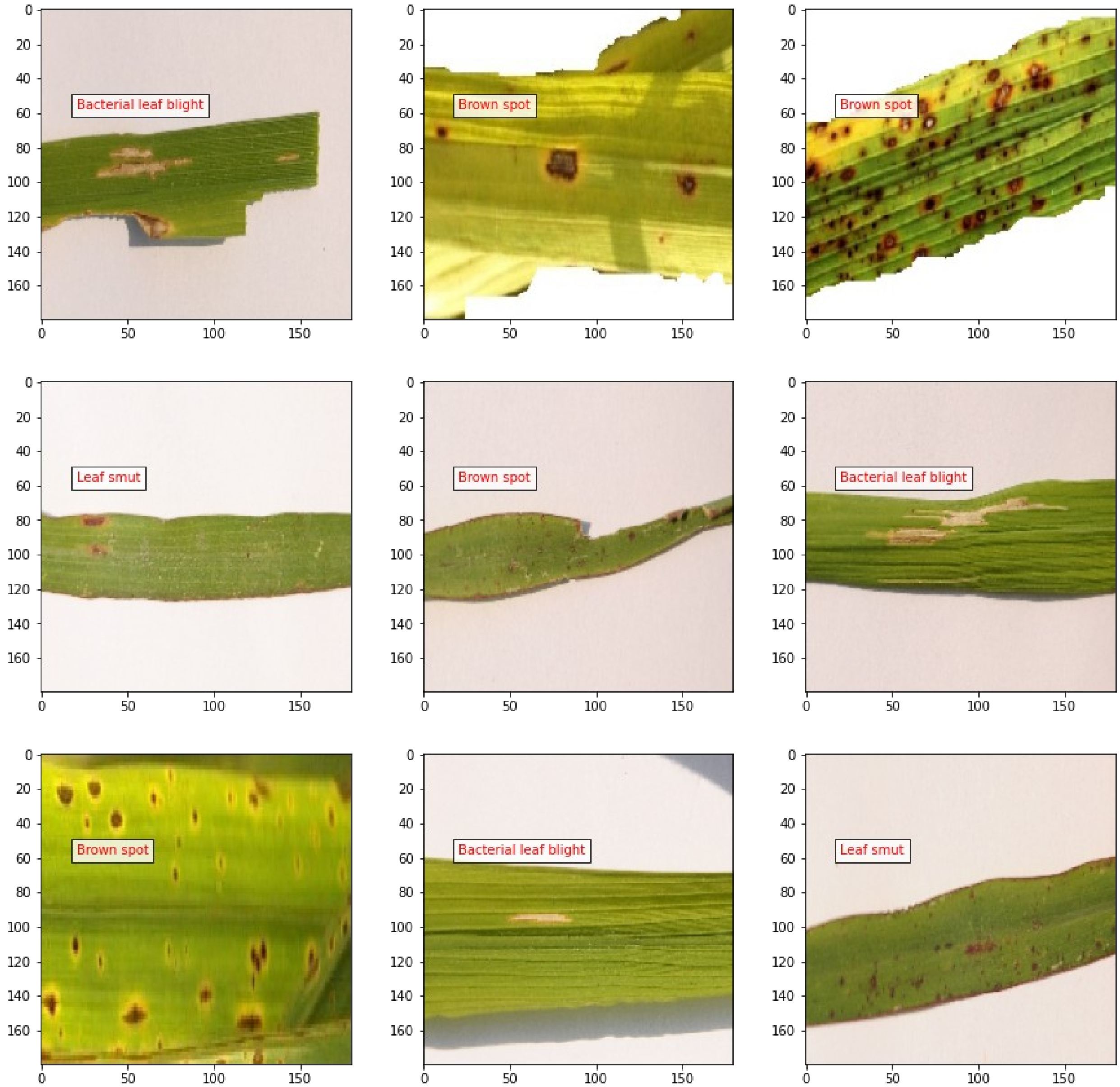
```
1/1 [==============================] - 0s 329ms/step - loss: 0.5341 -
accuracy: 0.9167
```

```
[0.5340743660926819, 0.9166666865348816]
```

- Here the loss is 0.53 and the accuracy of the model is 0.91 percent means 91%.

VISUALISE THE PREDICTION OF MODEL

```
# Visualise the prediction of the model
imgs, labels = next(test_generator)
fig =plt.figure(figsize=(15,15))
columns = 3
rows = 3
for i in range(columns*rows):
    fig.add_subplot(rows, columns, i+1)
    img_t = np.expand_dims(imgs[i],axis=0)
    prediction = model.predict(img_t)
    idx = prediction[0].tolist().index(max(prediction[0]))
    plt.text(20,58,
class_names[idx],color='red',fontsize=10,bbox=dict(facecolor='white',a
lpha=0.8))
    plt.imshow(imgs[i])
```


Bacterial leaf blight
Brown spot
Brown spot
Leaf smut
Brown spot
Bacterial leaf blight
Brown spot
Bacterial leaf blight
Leaf smut