In [ ]:
```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt, numpy as np
```

In [ ]:
```
# creating constants
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS = 3
EPOCHS = 30
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

In [ ]:
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/dataset',
    shuffle = True,
    image_size = (IMAGE_SIZE, IMAGE_SIZE),
    batch_size = BATCH_SIZE)
```

```
Found 2152 files belonging to 3 classes.
```

Making a list of the classes present in the dataset

In [ ]:
```
class_names = dataset.class_names
class_names
```

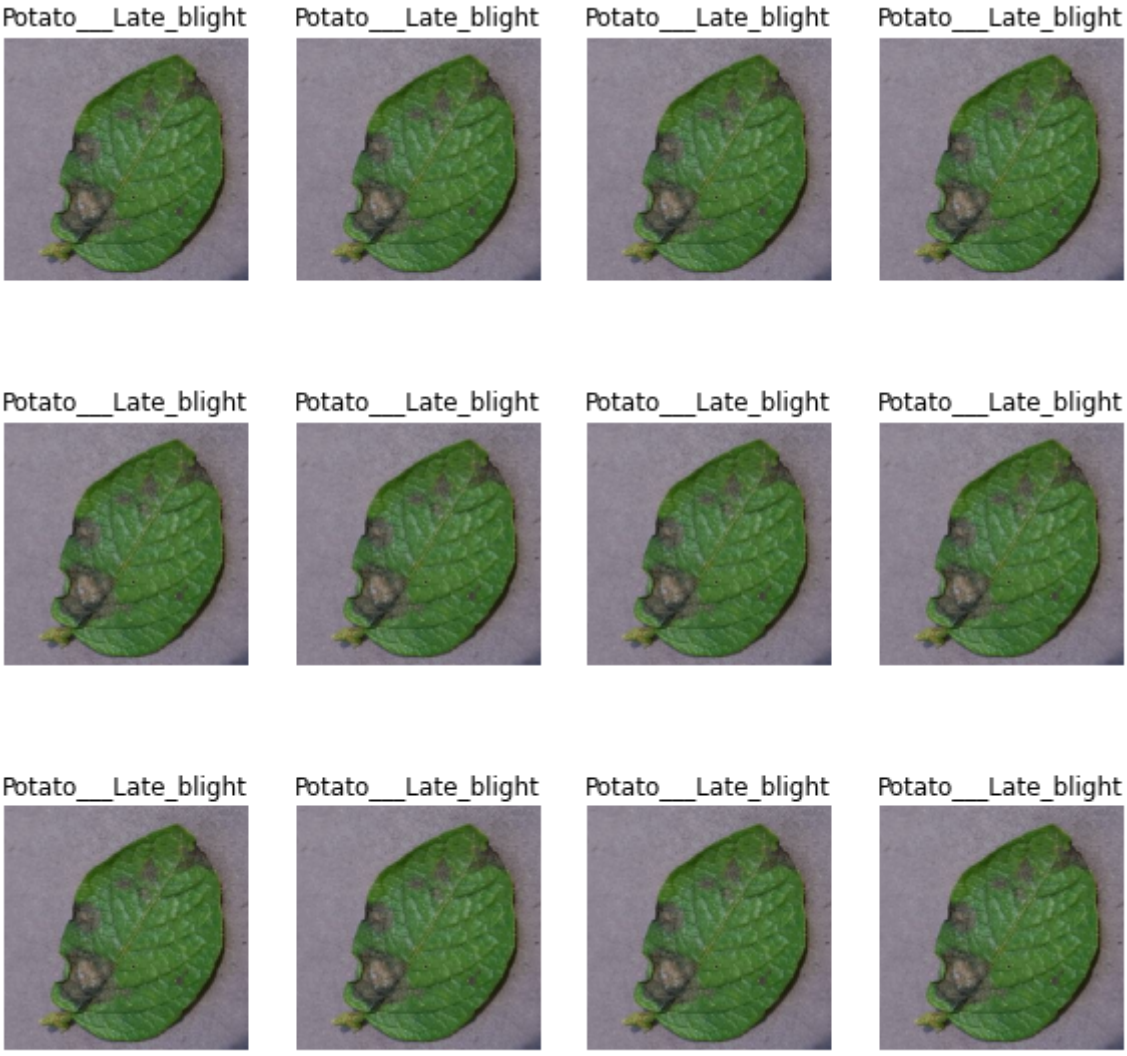
Out[ ]:
```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

In [ ]:
```
len(dataset)
# every element in the dataset is a batch of 32 images, so 68*32 = 2176 ima
```

Out[ ]:
```
68
```

In [ ]:
```
plt.figure(figsize = (10 , 10))
for image_batch, label_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i+1)
        plt.imshow(image_batch[0].numpy().astype('uint8'))
        plt.axis('off')
        plt.title(class_names[label_batch[0]])
    print('Image batch shape: ', image_batch.shape)
    print('Label batch', label_batch.numpy())
```

```
Image batch shape:  (32, 256, 256, 3)
Label batch [1 0 1 1 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 0 2 0 1 0 1
1]
```



In [ ]:
```
def partition_dataset(df ,train_split = 0.8, test_split = 0.1, valid_ds = 0
    if shuffle:
        df = df.shuffle(shuffle_size, seed = 12)
    length = len(df)
    train_l = int(length*train_split)
    test_l = int(length*test_split)
    valid_l = int(length*valid_ds)
    train_df = df.take(train_l)
    valid_df = df.skip(train_l).take(valid_l)
    test_df = df.skip(train_l+valid_l)
    return train_df, valid_df, test_df
```

In [ ]:
```
train_df, valid_df, test_df = partition_dataset(dataset)
```

In [ ]:
```
train_df = train_df.cache().shuffle(1000).prefetch(buffer_size = tf.data.AU
valid_df = valid_df.cache().shuffle(1000).prefetch(buffer_size = tf.data.AU
test_df = test_df.cache().shuffle(1000).prefetch(buffer_size = tf.data.AUTO
```

image preprocessing - creating layers for preprocessing

In [ ]:
```
# scaling the numpy array
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255) # scaling the valu
])
```

In [ ]:
```
# data augmentation
data_augmentation = tf.keras.Sequential([
```

```
    layers.experimental.preprocessing.RandomFlip('horizontal_and_vertical')
    layers.experimental.preprocessing.RandomRotation(0.2),
])
```

In [ ]:
```
# convolutional neural network
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    data_augmentation,

    layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (BATCH_SIZE
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(n_classes, activation = 'softmax')


])

model.build(input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS))
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0

 sequential_1 (Sequential)   (32, 256, 256, 3)         0

 conv2d (Conv2D)             (32, 254, 254, 32)        896

 max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
 )

 conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

 max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
 2D)

 conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928

 max_pooling2d_2 (MaxPooling  (32, 30, 30, 64)         0
 2D)

 conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928

 max_pooling2d_3 (MaxPooling  (32, 14, 14, 64)         0
 2D)

 conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928

 max_pooling2d_4 (MaxPooling  (32, 6, 6, 64)           0
 2D)

 conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928

 max_pooling2d_5 (MaxPooling  (32, 2, 2, 64)           0
 2D)

 flatten (Flatten)           (32, 256)                 0

 dense (Dense)               (32, 64)                  16448

 dense_1 (Dense)             (32, 3)                   195

=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model.compile(
    optimizer = 'adam',
    loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = Fals
    metrics = ['accuracy']

)
```

In [ ]:
```
history = model.fit(
    train_df,
    epochs = 30,
    batch_size=BATCH_SIZE,
    verbose = 1,
    validation_data = valid_df
```

```
)
```

```
Epoch 1/30
54/54 [==============================] - 114s 146ms/step - loss: 0.9015 - a
ccuracy: 0.5179 - val_loss: 0.8625 - val_accuracy: 0.5052
Epoch 2/30
54/54 [==============================] - 4s 72ms/step - loss: 0.7211 - accu
racy: 0.6748 - val_loss: 0.6262 - val_accuracy: 0.7396
Epoch 3/30
54/54 [==============================] - 4s 71ms/step - loss: 0.4448 - accu
racy: 0.8096 - val_loss: 0.3381 - val_accuracy: 0.8542
Epoch 4/30
54/54 [==============================] - 4s 71ms/step - loss: 0.2920 - accu
racy: 0.8866 - val_loss: 0.2995 - val_accuracy: 0.8802
Epoch 5/30
54/54 [==============================] - 4s 71ms/step - loss: 0.2592 - accu
racy: 0.8924 - val_loss: 0.2735 - val_accuracy: 0.8958
Epoch 6/30
54/54 [==============================] - 4s 71ms/step - loss: 0.2273 - accu
racy: 0.9016 - val_loss: 0.2034 - val_accuracy: 0.9062
Epoch 7/30
54/54 [==============================] - 4s 71ms/step - loss: 0.1969 - accu
racy: 0.9172 - val_loss: 0.2095 - val_accuracy: 0.9167
Epoch 8/30
54/54 [==============================] - 4s 71ms/step - loss: 0.1757 - accu
racy: 0.9334 - val_loss: 0.2278 - val_accuracy: 0.9115
Epoch 9/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1821 - accu
racy: 0.9306 - val_loss: 0.2667 - val_accuracy: 0.8802
Epoch 10/30
54/54 [==============================] - 4s 71ms/step - loss: 0.1914 - accu
racy: 0.9300 - val_loss: 0.2111 - val_accuracy: 0.9219
Epoch 11/30
54/54 [==============================] - 4s 71ms/step - loss: 0.1505 - accu
racy: 0.9433 - val_loss: 0.2141 - val_accuracy: 0.9427
Epoch 12/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1336 - accu
racy: 0.9520 - val_loss: 0.1609 - val_accuracy: 0.9375
Epoch 13/30
54/54 [==============================] - 4s 76ms/step - loss: 0.1315 - accu
racy: 0.9502 - val_loss: 0.1356 - val_accuracy: 0.9479
Epoch 14/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1296 - accu
racy: 0.9497 - val_loss: 0.1619 - val_accuracy: 0.9271
Epoch 15/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1311 - accu
racy: 0.9508 - val_loss: 0.1293 - val_accuracy: 0.9635
Epoch 16/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1174 - accu
racy: 0.9595 - val_loss: 0.1978 - val_accuracy: 0.9479
Epoch 17/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1171 - accu
racy: 0.9595 - val_loss: 0.1049 - val_accuracy: 0.9583
Epoch 18/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1078 - accu
racy: 0.9601 - val_loss: 0.0914 - val_accuracy: 0.9531
Epoch 19/30
54/54 [==============================] - 4s 72ms/step - loss: 0.0897 - accu
racy: 0.9670 - val_loss: 0.1611 - val_accuracy: 0.9583
Epoch 20/30
54/54 [==============================] - 4s 72ms/step - loss: 0.0988 - accu
racy: 0.9624 - val_loss: 0.1201 - val_accuracy: 0.9583
Epoch 21/30
54/54 [==============================] - 4s 72ms/step - loss: 0.1017 - accu
racy: 0.9595 - val_loss: 0.2379 - val_accuracy: 0.8958
Epoch 22/30
```

```
54/54 [==============================] - 4s 73ms/step - loss: 0.0952 - accu
racy: 0.9653 - val_loss: 0.1434 - val_accuracy: 0.9479
Epoch 23/30
54/54 [==============================] - 4s 73ms/step - loss: 0.0684 - accu
racy: 0.9769 - val_loss: 0.1720 - val_accuracy: 0.9531
Epoch 24/30
54/54 [==============================] - 4s 73ms/step - loss: 0.0762 - accu
racy: 0.9705 - val_loss: 0.1320 - val_accuracy: 0.9531
Epoch 25/30
54/54 [==============================] - 4s 76ms/step - loss: 0.0907 - accu
racy: 0.9664 - val_loss: 0.2261 - val_accuracy: 0.9115
Epoch 26/30
54/54 [==============================] - 5s 85ms/step - loss: 0.0782 - accu
racy: 0.9757 - val_loss: 0.0978 - val_accuracy: 0.9583
Epoch 27/30
54/54 [==============================] - 4s 75ms/step - loss: 0.0686 - accu
racy: 0.9757 - val_loss: 0.1071 - val_accuracy: 0.9688
Epoch 28/30
54/54 [==============================] - 4s 72ms/step - loss: 0.0982 - accu
racy: 0.9612 - val_loss: 0.1802 - val_accuracy: 0.9167
Epoch 29/30
54/54 [==============================] - 4s 72ms/step - loss: 0.0547 - accu
racy: 0.9803 - val_loss: 0.0573 - val_accuracy: 0.9792
Epoch 30/30
54/54 [==============================] - 4s 72ms/step - loss: 0.0487 - accu
racy: 0.9838 - val_loss: 0.0447 - val_accuracy: 0.9792
```

In [ ]:
```
model.evaluate(test_df)
```

```
8/8 [==============================] - 4s 33ms/step - loss: 0.0574 - accura
cy: 0.9805
```

Out[ ]:
```
[0.05741656571626663, 0.98046875]
```

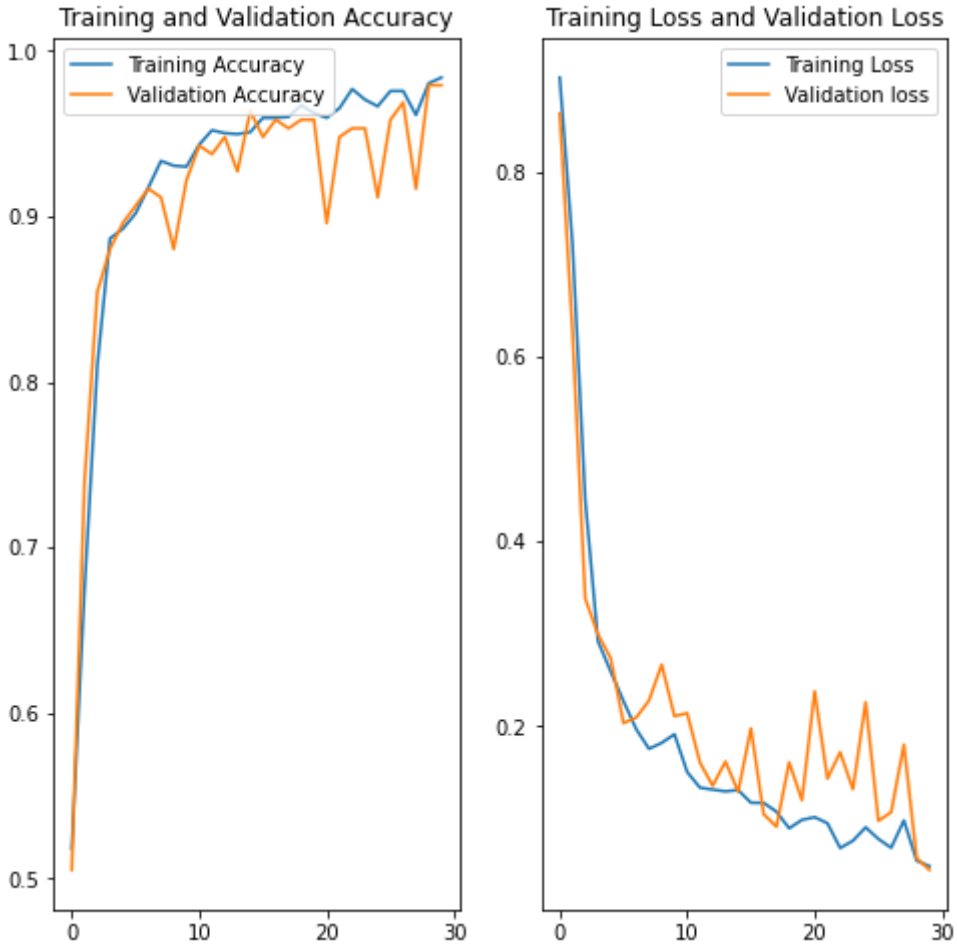
In [ ]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [ ]:
```
plt.figure(figsize = (8,8))
plt.subplot(1,2,1)

plt.plot(range(EPOCHS), acc, label = 'Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'upper left')
plt.title('Training and Validation Accuracy')

plt.subplot(1,2, 2)
plt.plot(range(EPOCHS), loss, label = 'Training Loss')
plt.plot(range(EPOCHS), val_loss, label = 'Validation loss')
plt.legend(loc = 'upper right')
plt.title('Training Loss and Validation Loss')
```

Out[ ]:
```
Text(0.5, 1.0, 'Training Loss and Validation Loss')
```

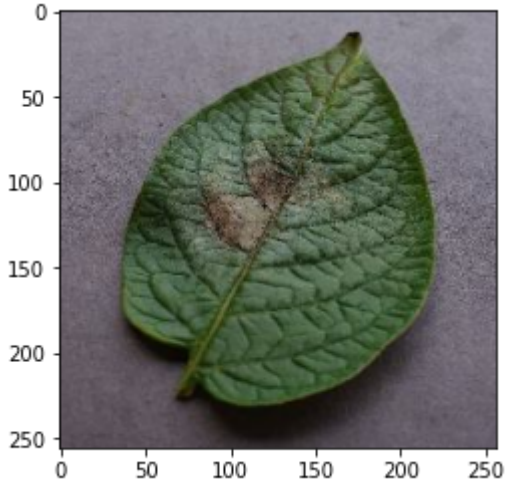


PREDICTING

In [ ]:
```
for images_batch, labels_batch in test_df.take(1):
  print(len(images_batch), len(labels_batch))
  first_image = images_batch[0].numpy().astype('uint8')
  first_label = labels_batch[0].numpy()

  print('Actual image label vs predicted label')
  plt.imshow(first_image)
  print('Actual label : ', class_names[first_label])
  print('predicted label :', class_names[np.argmax(model.predict(images_bat
```

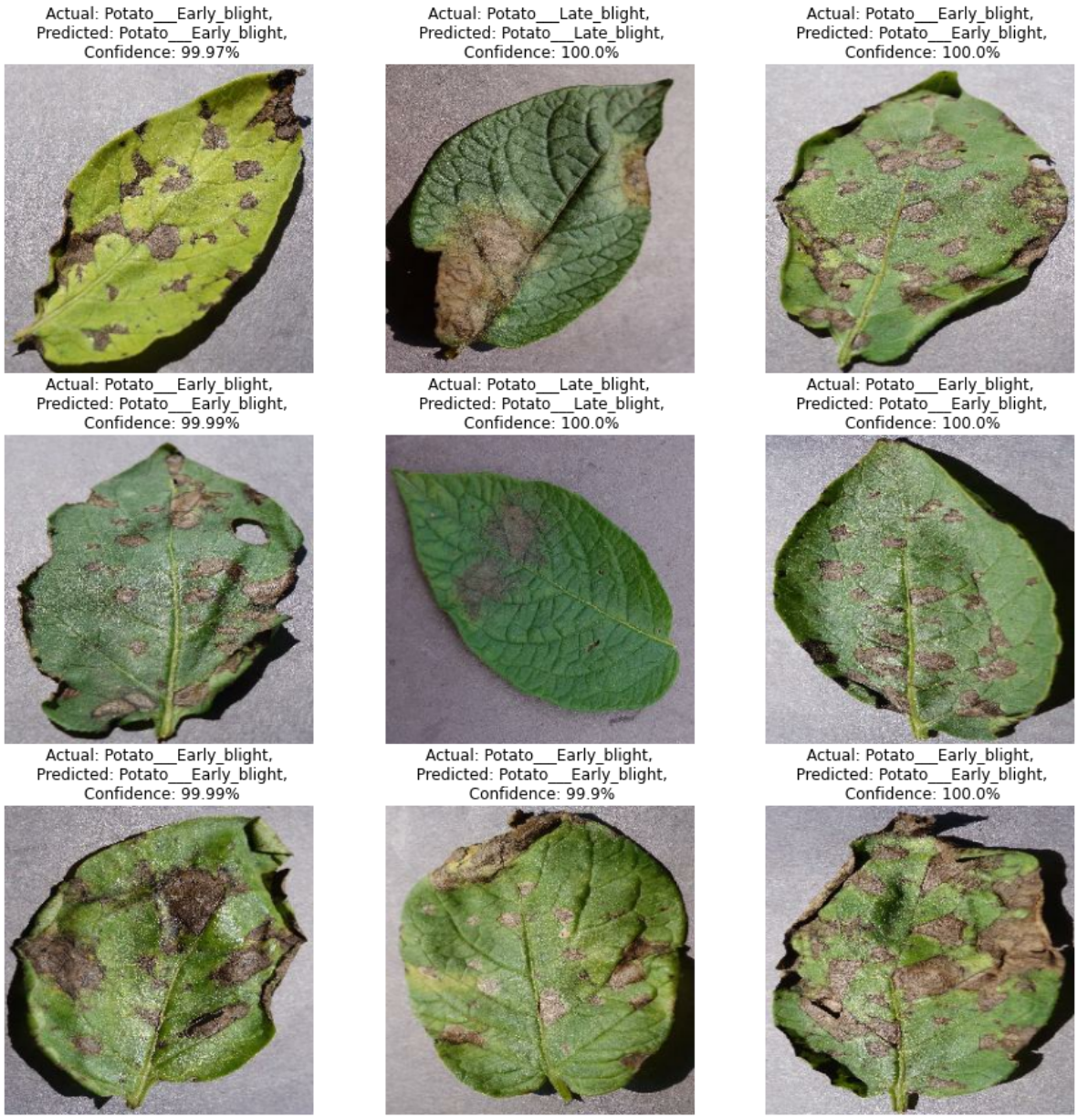
```
32 32
Actual image label vs predicted label
Actual label :  Potato___Late_blight
predicted label : Potato___Late_blight
```



In [ ]:
```
def predict(model, img):
  img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
  img_array = tf.expand_dims(img_array, 0)

  predictions =  model.predict(img_array)
  predicted_class = class_names[np.argmax(predictions[0])]
  confidence = round(100*(np.max(predictions[0])), 2)
  return predicted_class,confidence
```

In [ ]:
```
plt.figure(figsize = (15,15))
for images, labels in test_df.take(1):
  for i in range(9):
    ax = plt.subplot(3,3,i+1)
    plt.imshow(images[i].numpy().astype('uint8'))
    plt.axis('off')
    predicted_class, confidence = predict(model, images[i].numpy())
    actual_class = class_names[labels[i]]
    plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class},\n C
```



In [ ]:
```
model_version = 1
model.save(f"../models/{model_version}")
```

```
INFO:tensorflow:Assets written to: ../models/1/assets
```