Potato_Leaf_Disease

In [1]:
```
# import libraries
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pathlib
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' # to disable all debugging logs
```

In [2]:
```
# importing dataset
dir = os.listdir(r'C:\Users\misal\OneDrive\Desktop\potato1\archive\traning\PlantVillage')
for filenames in dir:
    print(filenames)
```

```
Potato___Early_blight
Potato___healthy
Potato___Late_blight
```

In [55]:
```
#Global initialization of some imp variables
Image_Size = 256
Batch_Size = 32
Epochs = 40
```

In [56]:
```
img_height, img_width = 32, 32
batch_size = 20

train_ds = tf.keras.utils.image_dataset_from_directory(
    "C:\POTATO DISEASE\PlantVillage",
    image_size = (img_height, img_width),
    batch_size = batch_size
)
val_ds = tf.keras.utils.image_dataset_from_directory(
    "C:\POTATO DISEASE\PlantVillage",
    image_size = (img_height, img_width),
    batch_size = batch_size
)
test_ds = tf.keras.utils.image_dataset_from_directory(
    "C:\POTATO DISEASE\PlantVillage",
    image_size = (img_height, img_width),
    batch_size = batch_size
)
```

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

In [57]:
```
#Folders(classes) in 'Dataset' directory
class_name = train_ds.class_names
class_name
```

Out[57]:
```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

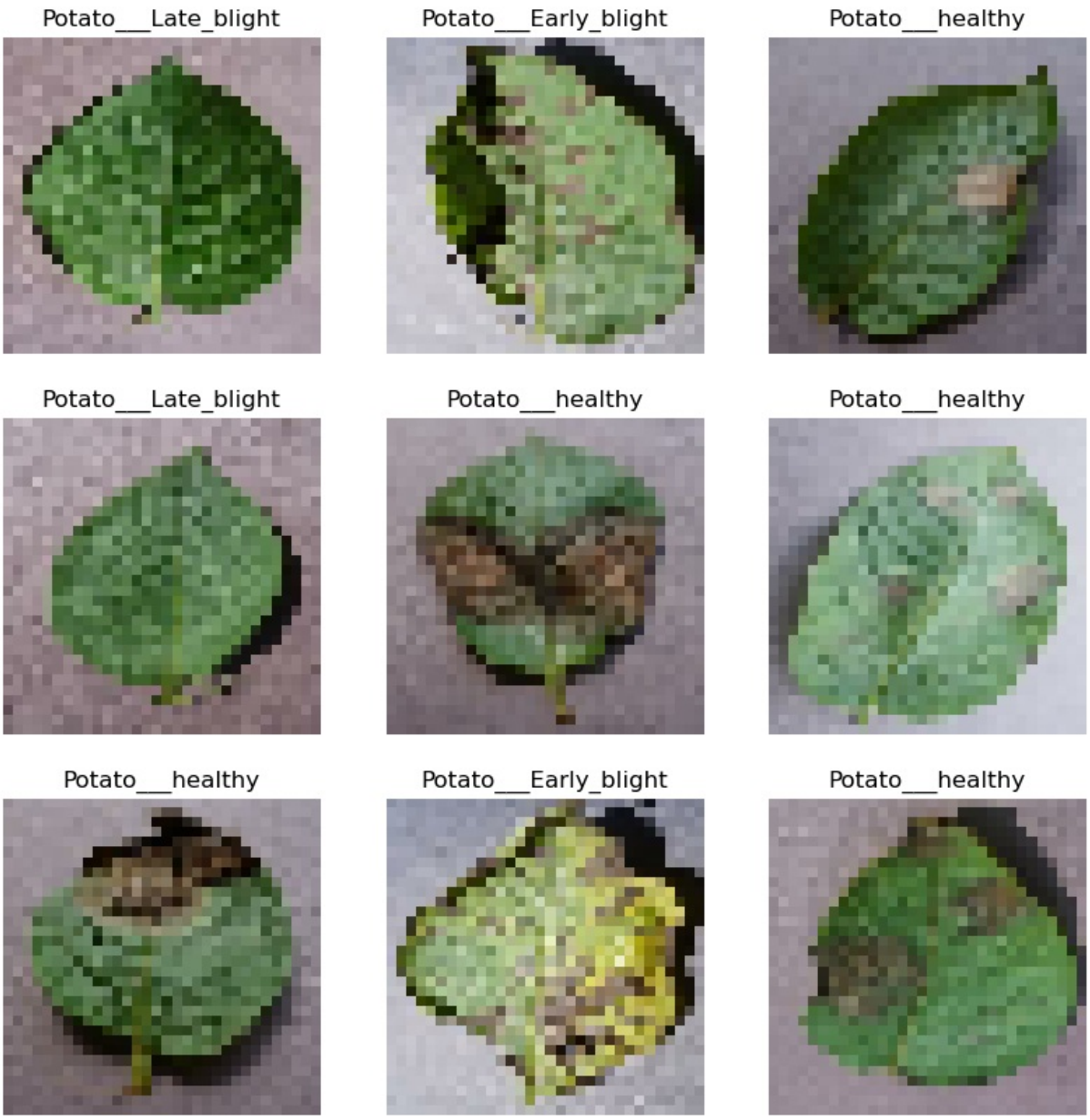
In [58]:
```
len(train_ds) # Number of Batches = (total number of files belonging to all classes / Batch_Size)
```

Out[58]:
```
108
```

In [59]:
```
print(train_ds) #prints Elements in dataset: here 1st element is image and 2nd index of that image.
```

```
<BatchDataset element_spec=(TensorSpec(shape=(None, 32, 32, 3), dtype=tf.float32, name=None), TensorSpec(shape=(
None,), dtype=tf.int32, name=None))>
```

In [60]:
```
class_names = ["Potato___Early_blight", "Potato___healthy", "Potato___Late_blight"]
# Plotting the image
plt.figure(figsize=(10,10))
# dataset.take(count) : Creates a Dataset with at most 'count' elements(batch) from the dataset
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)   # many plots at a time =>subpots

    plt.imshow(images[i].numpy().astype("uint8"))  #converting all data of image into numpy and than to intiger
    plt.title(class_names[labels[i]])   # title of the class_name of image
    plt.axis("off")  # Hide the values of graph
```



Data Pre-processing

In [61]:
```
# Image Preprocessing : Rescaling and Resizing
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(Image_Size, Image_Size),
    layers.experimental.preprocessing.Rescaling(1./255)
])
```

In [62]:
```
# Data augmentation by flipping and rotating existing images
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip(mode="horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(factor = 0.5)
])
```

Model Building

Creating Convolution layer

In [38]:
```
model = tf.keras.Sequential(
    [
     tf.keras.layers.Rescaling(1./255),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Flatten(),
     tf.keras.layers.Dense(128, activation="relu"),
     tf.keras.layers.Dense(3)
```

```
    ]
)
```

In [50]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_2 (Rescaling)     (None, 32, 32, 3)         0

 conv2d_6 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_6 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 13, 13, 32)        9248

 max_pooling2d_7 (MaxPooling  (None, 6, 6, 32)         0
 2D)

 conv2d_8 (Conv2D)           (None, 4, 4, 32)          9248

 max_pooling2d_8 (MaxPooling  (None, 2, 2, 32)         0
 2D)

 flatten_2 (Flatten)         (None, 128)               0

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 3)                 387

=================================================================
Total params: 36,291
Trainable params: 36,291
Non-trainable params: 0
_________________________________________________________________
```

In [39]:
```
# Optimizing the model 'SparseCategoricalCrossentropy'=>as there are many categorical classes of data
model.compile(
    optimizer="adam",
    loss=tf.losses.SparseCategoricalCrossentropy(from_logits = True),
    metrics=['accuracy']
)
```

In [40]:
```
 #Fit the model with training data and also pass validation data
    history = model.fit(
    train_ds,
    validation_data = val_ds,
    epochs = Epochs
)
```

```
Epoch 1/40
108/108 [==============================] - 2s 18ms/step - loss: 0.8091 - accuracy: 0.6101 - val_loss: 0.5719 - v
al_accuracy: 0.7988
Epoch 2/40
108/108 [==============================] - 2s 18ms/step - loss: 0.5209 - accuracy: 0.7849 - val_loss: 0.4221 - v
al_accuracy: 0.8155
Epoch 3/40
108/108 [==============================] - 2s 18ms/step - loss: 0.3810 - accuracy: 0.8429 - val_loss: 0.2903 - v
al_accuracy: 0.9024
Epoch 4/40
108/108 [==============================] - 2s 18ms/step - loss: 0.2667 - accuracy: 0.8913 - val_loss: 0.2378 - v
al_accuracy: 0.9112
Epoch 5/40
108/108 [==============================] - 2s 18ms/step - loss: 0.2211 - accuracy: 0.9136 - val_loss: 0.1591 - v
al_accuracy: 0.9382
Epoch 6/40
108/108 [==============================] - 2s 19ms/step - loss: 0.1446 - accuracy: 0.9470 - val_loss: 0.1065 - v
al_accuracy: 0.9628
Epoch 7/40
108/108 [==============================] - 2s 17ms/step - loss: 0.1299 - accuracy: 0.9507 - val_loss: 0.1882 - v
al_accuracy: 0.9261
Epoch 8/40
108/108 [==============================] - 2s 19ms/step - loss: 0.1407 - accuracy: 0.9428 - val_loss: 0.0776 - v
al_accuracy: 0.9707
Epoch 9/40
108/108 [==============================] - 2s 21ms/step - loss: 0.0907 - accuracy: 0.9684 - val_loss: 0.0705 - v
al_accuracy: 0.9800
Epoch 10/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0927 - accuracy: 0.9633 - val_loss: 0.0723 - v
al_accuracy: 0.9726
Epoch 11/40
108/108 [==============================] - 2s 17ms/step - loss: 0.0756 - accuracy: 0.9721 - val_loss: 0.0358 - v
```

```
al_accuracy: 0.9935
Epoch 12/40
108/108 [==============================] - 2s 18ms/step - loss: 0.0562 - accuracy: 0.9791 - val_loss: 0.0537 - v
al_accuracy: 0.9851
Epoch 13/40
108/108 [==============================] - 2s 17ms/step - loss: 0.0705 - accuracy: 0.9749 - val_loss: 0.0320 - v
al_accuracy: 0.9940
Epoch 14/40
108/108 [==============================] - 2s 18ms/step - loss: 0.0431 - accuracy: 0.9833 - val_loss: 0.0358 - v
al_accuracy: 0.9875
Epoch 15/40
108/108 [==============================] - 2s 18ms/step - loss: 0.0497 - accuracy: 0.9823 - val_loss: 0.0424 - v
al_accuracy: 0.9875
Epoch 16/40
108/108 [==============================] - 2s 20ms/step - loss: 0.0357 - accuracy: 0.9888 - val_loss: 0.0642 - v
al_accuracy: 0.9726
Epoch 17/40
108/108 [==============================] - 2s 18ms/step - loss: 0.0275 - accuracy: 0.9902 - val_loss: 0.0129 - v
al_accuracy: 0.9991
Epoch 18/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0296 - accuracy: 0.9888 - val_loss: 0.0112 - v
al_accuracy: 0.9986
Epoch 19/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0181 - accuracy: 0.9954 - val_loss: 0.0112 - v
al_accuracy: 0.9981
Epoch 20/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0614 - accuracy: 0.9758 - val_loss: 0.0163 - v
al_accuracy: 0.9944
Epoch 21/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0173 - accuracy: 0.9954 - val_loss: 0.0060 - v
al_accuracy: 0.9995
Epoch 22/40
108/108 [==============================] - 2s 18ms/step - loss: 0.0060 - accuracy: 1.0000 - val_loss: 0.0032 - v
al_accuracy: 1.0000
Epoch 23/40
108/108 [==============================] - 2s 17ms/step - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.0043 - v
al_accuracy: 1.0000
Epoch 24/40
108/108 [==============================] - 2s 18ms/step - loss: 0.0093 - accuracy: 0.9977 - val_loss: 0.0054 - v
al_accuracy: 0.9995
Epoch 25/40
108/108 [==============================] - 2s 20ms/step - loss: 0.0123 - accuracy: 0.9963 - val_loss: 0.0049 - v
al_accuracy: 0.9995
Epoch 26/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0359 - accuracy: 0.9884 - val_loss: 0.0186 - v
al_accuracy: 0.9921
Epoch 27/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0125 - accuracy: 0.9967 - val_loss: 0.0019 - v
al_accuracy: 1.0000
Epoch 28/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0020 - v
al_accuracy: 1.0000
Epoch 29/40
108/108 [==============================] - 2s 19ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0011 - v
al_accuracy: 1.0000
Epoch 30/40
108/108 [==============================] - 2s 18ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 8.0997e-04
- val_accuracy: 1.0000
Epoch 31/40
108/108 [==============================] - 2s 18ms/step - loss: 8.6592e-04 - accuracy: 1.0000 - val_loss: 0.0011
- val_accuracy: 1.0000
Epoch 32/40
108/108 [==============================] - 2s 21ms/step - loss: 6.4779e-04 - accuracy: 1.0000 - val_loss: 5.3865
e-04 - val_accuracy: 1.0000
Epoch 33/40
108/108 [==============================] - 2s 19ms/step - loss: 5.5457e-04 - accuracy: 1.0000 - val_loss: 5.7004
e-04 - val_accuracy: 1.0000
Epoch 34/40
108/108 [==============================] - 2s 19ms/step - loss: 4.5760e-04 - accuracy: 1.0000 - val_loss: 4.3379
e-04 - val_accuracy: 1.0000
Epoch 35/40
108/108 [==============================] - 2s 19ms/step - loss: 4.6593e-04 - accuracy: 1.0000 - val_loss: 4.6884
e-04 - val_accuracy: 1.0000
Epoch 36/40
108/108 [==============================] - 2s 20ms/step - loss: 3.7683e-04 - accuracy: 1.0000 - val_loss: 3.7482
e-04 - val_accuracy: 1.0000
Epoch 37/40
108/108 [==============================] - 2s 19ms/step - loss: 3.4978e-04 - accuracy: 1.0000 - val_loss: 3.2144
e-04 - val_accuracy: 1.0000
Epoch 38/40
108/108 [==============================] - 2s 18ms/step - loss: 3.1005e-04 - accuracy: 1.0000 - val_loss: 3.0542
e-04 - val_accuracy: 1.0000
Epoch 39/40
```

```
108/108 [==============================] - 2s 19ms/step - loss: 2.8202e-04 - accuracy: 1.0000 - val_loss: 2.5608
e-04 - val_accuracy: 1.0000
Epoch 40/40
108/108 [==============================] - 2s 19ms/step - loss: 2.5675e-04 - accuracy: 1.0000 - val_loss: 2.3828
e-04 - val_accuracy: 1.0000
```

In [41]:
```
scores = model.evaluate(test_ds)
```

```
108/108 [==============================] - 1s 5ms/step - loss: 2.3828e-04 - accuracy: 1.0000
```

In [42]:
```
history
```

Out[42]:
```
<keras.callbacks.History at 0x1cb756a7108>
```

In [43]:
```
history.params
```

Out[43]:
```
{'verbose': 1, 'epochs': 40, 'steps': 108}
```

In [44]:
```
history.history.keys()
```

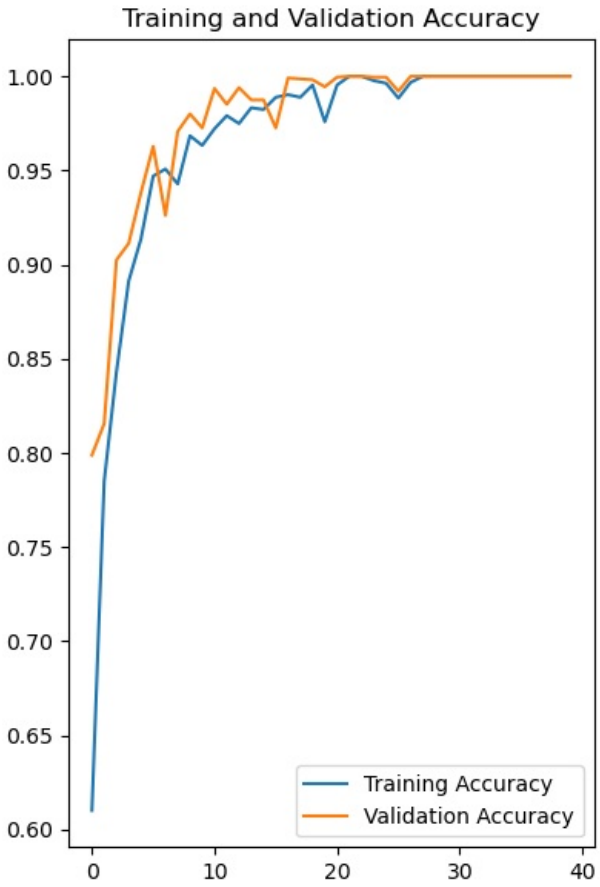
Out[44]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

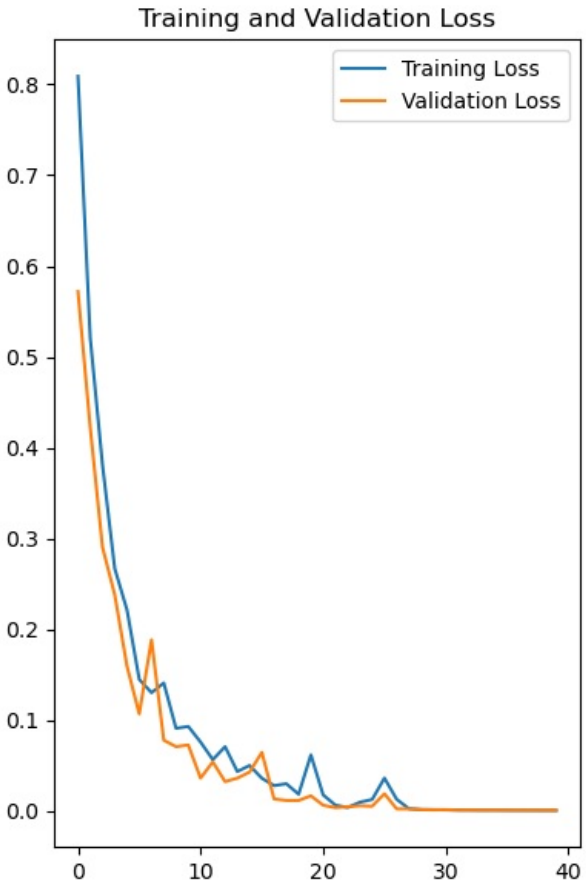
Analyzing the Output

In [45]:
```
# Getting the model history to analyse
train_loss = history.history['loss']
train_acc = history.history['accuracy']

val_loss = history.history['val_loss']
val_acc = history.history['val_accuracy']
```

In [46]:
```
#graphs for accuracy and loss of training and validation data
plt.figure(figsize = (15,15))
plt.subplot(2,3,1)
plt.plot(range(Epochs), train_acc, label = 'Training Accuracy')
plt.plot(range(Epochs), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2,3,2)
plt.plot(range(Epochs), train_loss, label = 'Training Loss')
plt.plot(range(Epochs), val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training and Validation Loss')
```

Out[46]:
```
Text(0.5, 1.0, 'Training and Validation Loss')
```





In [47]:
```
#plotting image
for batch_image, batch_label in train_ds.take(1):
    first_image = batch_image[0].numpy().astype('uint8')
    first_label = class_name[batch_label[0]]

    print('First Image of batch to predict :')
    plt.imshow(first_image)
    print('Actual label : ', first_label)

    batch_prediction = model.predict(batch_image)
    print('Predicted label : ', class_name[np.argmax(batch_prediction[0])])
    plt.axis('off')
```

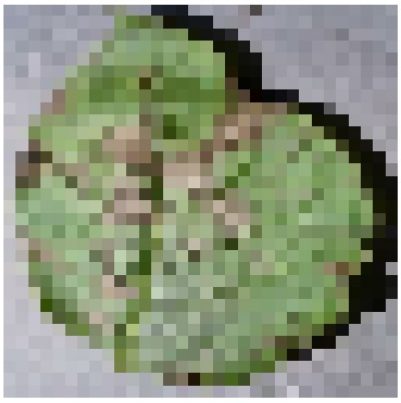
```
First Image of batch to predict :
Actual label :  Potato___healthy
1/1 [==============================] - 0s 67ms/step
Predicted label :  Potato___healthy
```

In [49]:
```python
# plotting batch of images with its actual label, predicted label and confidence

plt.figure(figsize = (16,16))
for batch_image, batch_label in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        image = batch_image[i].numpy().astype('uint8')
        label = class_name[batch_label[i]]

        plt.imshow(image)

        batch_prediction = model.predict(batch_image)
        predicted_class = class_name[np.argmax(batch_prediction[i])]
        confidence = round(np.max(batch_prediction[i]) * 100, 2)

        plt.title(f'Actual : {label},\n Prediction : {predicted_class},\n Confidence : {confidence}%')

        plt.axis('off')
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
```

Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 1800.31%

Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 2059.48%



Actual : Potato___healthy,
Prediction : Potato___healthy,
Confidence : 1420.04%

Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 783.06%

Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 1581.81%

Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 2729.04%

Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 2176.32%

Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 1479.45%

Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 1066.35%

In [65]:
```
model_version=1
model.save('C:\Users\misal\OneDrive\Desktop\potato1\archive\models.csv')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: C:\Users\misal\OneDrive\Desktop\potato1\archive\models.csv\assets
INFO:tensorflow:Assets written to: C:\Users\misal\OneDrive\Desktop\potato1\archive\models.csv\assets
```

The Model shows accuracy > 97%