In [25]:

```
#importing necessary libraries

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Input
import matplotlib.pyplot as plt
from PIL import Image
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense,Dropout,BatchNormalization
```

In [26]:

```
#to create a new sequential model object

model = Sequential()
```

In [27]:

```
# Firt Convolution layer

model.add(Conv2D(32, 5, 5, input_shape = (256, 256, 3), activation = 'relu'))
model.add(BatchNormalization())
```

In [28]:

```
# adding a Pooling layer

model.add(MaxPooling2D(pool_size = (2, 2)))
```

In [29]:

```
# Adding a second Convolutional & Pooling layer

model.add(Conv2D(64, 5, 5, activation = 'relu'))

model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(BatchNormalization())
```

In [30]:

```
# adding a Flattening layer

model.add(Flatten())
```

In [31]:

```
# Full connection

model.add(Dense(32, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Dense(1, activation = 'sigmoid'))
```

In [32]:

```
#  summary of the layers and parameters

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 51, 51, 32)        2432

 batch_normalization_3 (Batc  (None, 51, 51, 32)       128
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 25, 25, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 5, 5, 64)          51264

 max_pooling2d_4 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 batch_normalization_4 (Batc  (None, 2, 2, 64)         256
 hNormalization)

 flatten_2 (Flatten)         (None, 256)               0

 dense_2 (Dense)             (None, 32)                8224

 batch_normalization_5 (Batc  (None, 32)               128
 hNormalization)

 dropout_1 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 62,465
Trainable params: 62,209
Non-trainable params: 256
_________________________________________________________________
```

In [33]:

```
# Compile the model

model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

In [35]:

```
# Data augmentation

from keras.preprocessing.image import ImageDataGenerator

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,
    fill_mode = 'nearest')
```

In [36]:

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

In [38]:

```
# Loading training data

training_set = train_datagen.flow_from_directory('train',
                                                 target_size = (256, 256),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

```
Found 40 images belonging to 2 classes.
```

In [39]:

```
# Loading testing data

test_set = test_datagen.flow_from_directory('test',
                                            target_size = (256, 256),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

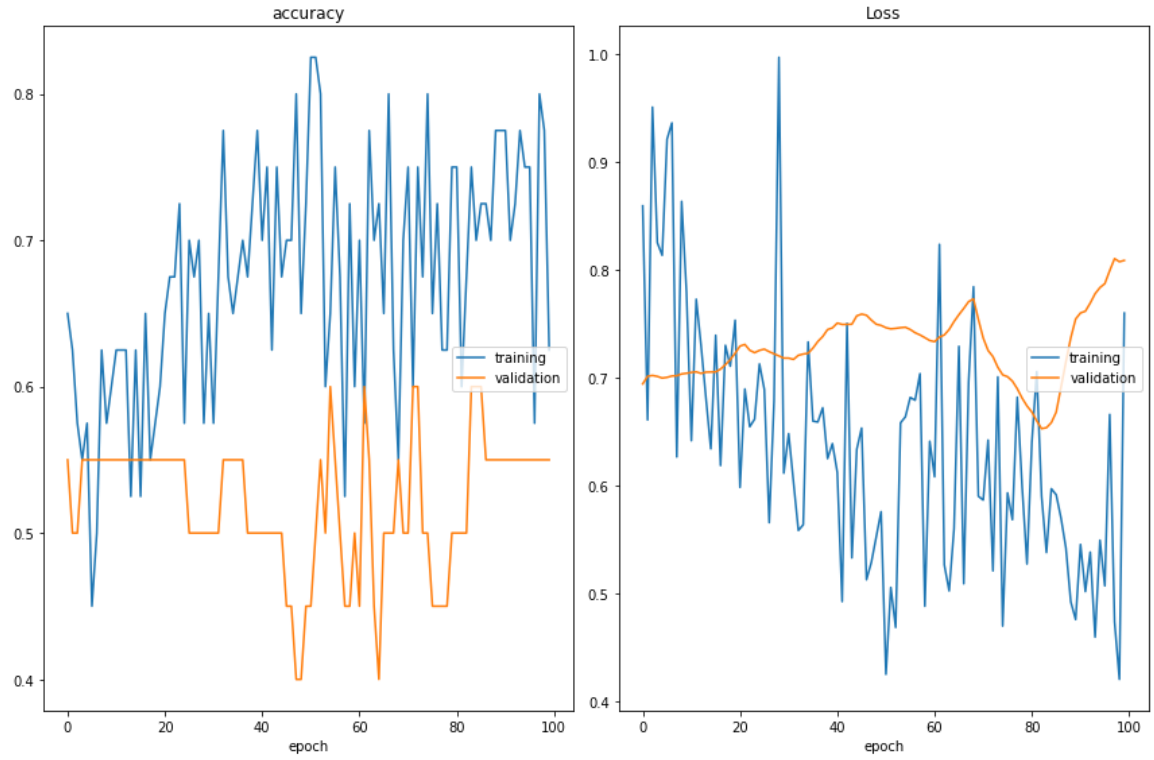
```
Found 20 images belonging to 2 classes.
```

In [40]:

```
from livelossplot import PlotLossesKerasTF
```

In [41]:

```
# Training model with 100 epochs

epochs_100 = model.fit(training_set,epochs=100,validation_data = test_set,callbacks=[Pl
otLossesKerasTF()])
```




```
accuracy
        training                 (min:    0.450, max:    0.825, cur:    0.
625)
        validation               (min:    0.400, max:    0.600, cur:    0.
550)
Loss
        training                 (min:    0.420, max:    0.997, cur:    0.
760)
        validation               (min:    0.653, max:    0.810, cur:    0.
809)
2/2 [==============================] - 1s 972ms/step - loss: 0.7599 - accu
racy: 0.6250 - val_loss: 0.8087 - val_accuracy: 0.5500
```

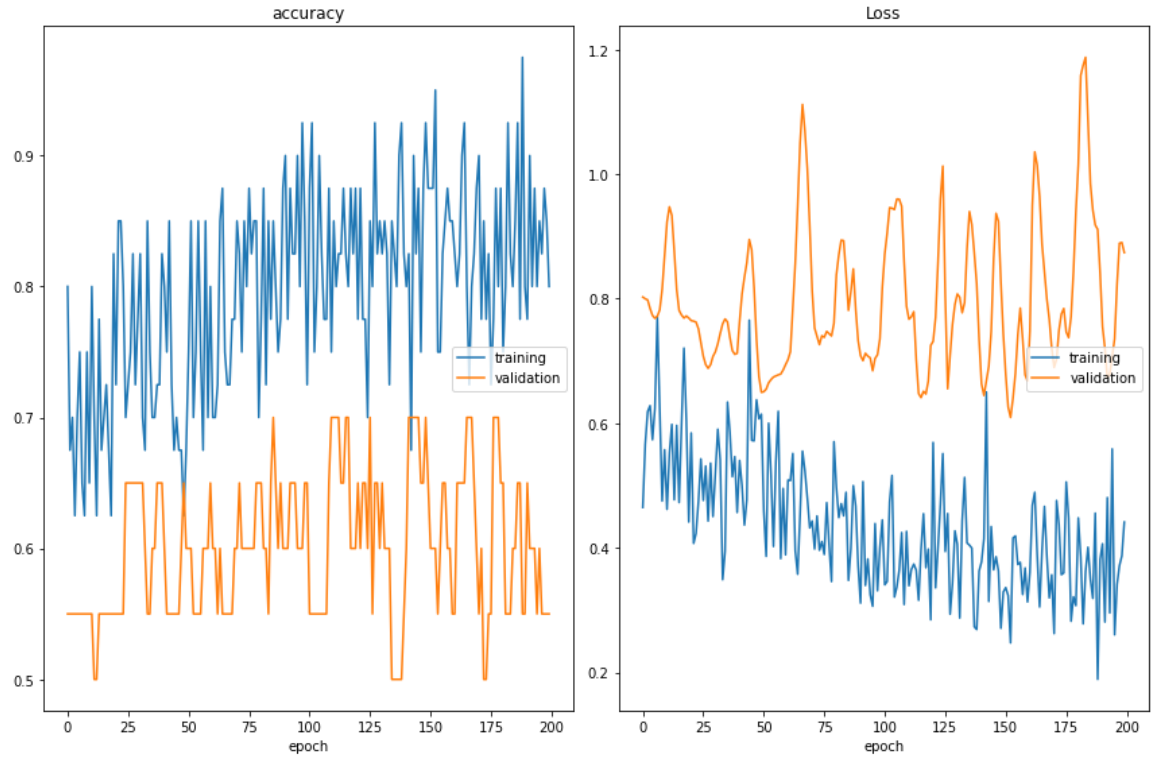
In [42]:

```
# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(test_set)

# Print the test accuracy and loss
print('Test accuracy:', test_acc)
print('Test loss:', test_loss)
```

```
1/1 [==============================] - 0s 109ms/step - loss: 0.8087 - accu
racy: 0.5500
Test accuracy: 0.550000011920929
Test loss: 0.8087366819381714
```

In [43]:

```
# Training model with 200 epochs

epochs_200 = model.fit(training_set,epochs=200,validation_data = test_set,callbacks=[Pl
otLossesKerasTF()])
```




```
accuracy
        training                 (min:    0.625, max:    0.975, cur:    0.
800)
        validation               (min:    0.500, max:    0.700, cur:    0.
550)
Loss
        training                 (min:    0.189, max:    0.771, cur:    0.
441)
        validation               (min:    0.609, max:    1.188, cur:    0.
874)
2/2 [==============================] - 1s 1s/step - loss: 0.4414 - accurac
y: 0.8000 - val_loss: 0.8741 - val_accuracy: 0.5500
```

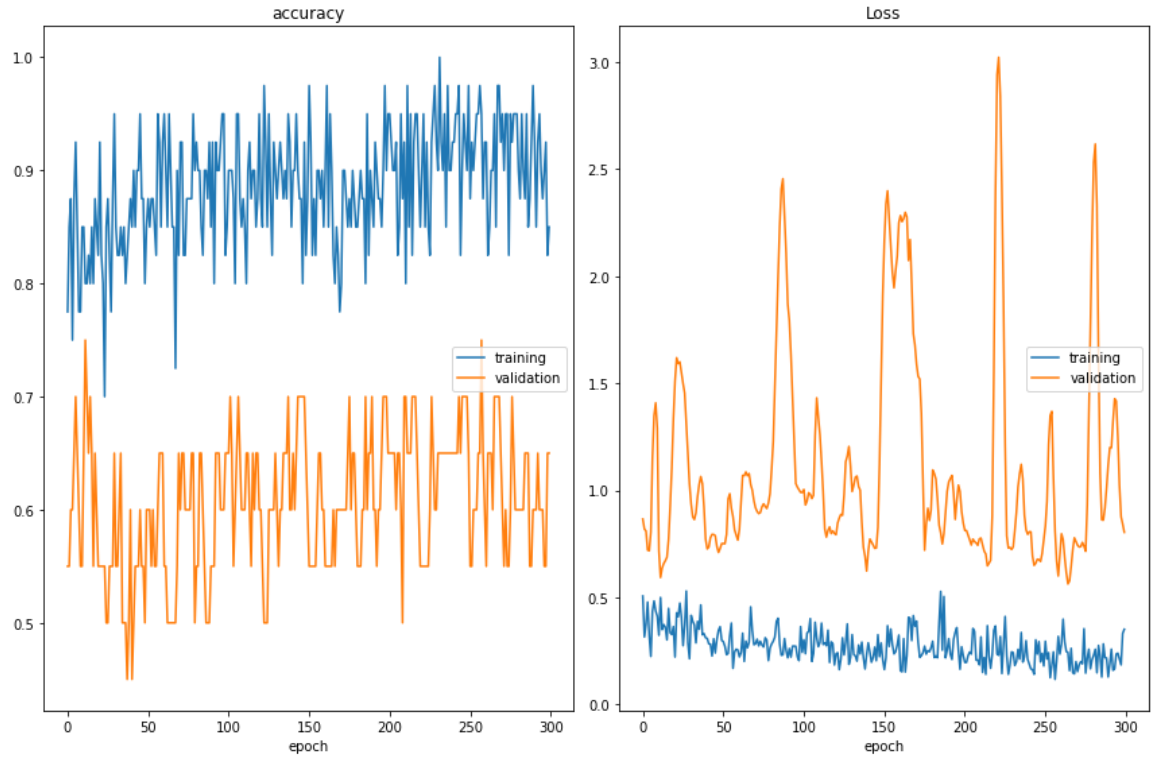
In [44]:

```
# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(test_set)

# Print the test accuracy and loss
print('Test accuracy:', test_acc)
print('Test loss:', test_loss)
```

```
1/1 [==============================] - 0s 133ms/step - loss: 0.8741 - accu
racy: 0.5500
Test accuracy: 0.550000011920929
Test loss: 0.8740615844726562
```

In [45]:

```
# Training model with 300 epochs

epochs_300 = model.fit(training_set,epochs=300,validation_data = test_set,callbacks=[Pl
otLossesKerasTF()])
```




```
accuracy
        training                 (min:    0.700, max:    1.000, cur:    0.
850)
        validation               (min:    0.450, max:    0.750, cur:    0.
650)
Loss
        training                 (min:    0.118, max:    0.530, cur:    0.
352)
        validation               (min:    0.563, max:    3.022, cur:    0.
805)
2/2 [==============================] - 1s 1s/step - loss: 0.3516 - accurac
y: 0.8500 - val_loss: 0.8047 - val_accuracy: 0.6500
```

In [46]:

```
# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(test_set)

# Print the test accuracy and loss
print('Test accuracy:', test_acc)
print('Test loss:', test_loss)
```

```
1/1 [==============================] - 0s 133ms/step - loss: 0.8047 - accu
racy: 0.6500
Test accuracy: 0.6499999761581421
Test loss: 0.8046718835830688
```