SPRINT-2
MODEL BUILDING

Date	17 NOV 2022
Team Id	PNT2022TMID01315
Project Name	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

Task:

Model Building is done using the following steps.

ADDING CNN LAYERS

```
In [12]: #MODEL BUILDING

 In [ ]: #Adding CNN Layers

In [13]: model = Sequential()

In [14]: model.add(Convolution2D(32,(3,3),input_shape = (64,64,3),activation = "relu"))

In [15]: model.add(MaxPooling2D(pool_size = (2,2)))

In [16]: model.add(Convolution2D(32,(3,3),activation='relu'))

In [17]: model.add(MaxPooling2D(pool_size=(2,2)))

In [18]: model.add(Flatten()) # ANN Input...
```

ADDING DENSE LAYERS

```
In [19]: #Adding Dense Layers

In [20]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [21]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [22]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [23]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [24]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))
```

ADDING OUTPUT LAYER

```
In [ ]: #Adding Output Layer
```

```
In [25]: model.add(Dense(units = 6,kernel_initializer = "random_uniform",activation = "softmax"))
```

The summary method is used to get the full information about the model and its layers.

```
In [26]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 128)               16512

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 6)                 774

=================================================================
Total params: 879,910
Trainable params: 879,910
Non-trainable params: 0
_________________________________________________________________
```

CONFIGURE THE LEARNING PROCESS

```
In [27]: model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

TRAIN THE MODEL

We will train the model using an image dataset.
Fit_generator function is used to train the deep learnming neural network.

```
In [30]: model.fit_generator(generator=x_train,steps_per_epoch = len(x_train), epochs=9, validation_data=x_test,validation_steps = len(x_t

C:\Users\LIKITHA S\AppData\Local\Temp\ipykernel_11640\788911318.py:1: UserWarning: `Model.fit_generator` is deprecated and will
be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(generator=x_train,steps_per_epoch = len(x_train), epochs=9, validation_data=x_test,validation_steps = len
(x_test))

Epoch 1/9
480/480 [==============================] - 79s 162ms/step - loss: 1.4184 - accuracy: 0.4806 - val_loss: 1.3712 - val_accuracy:
0.3927
Epoch 2/9
480/480 [==============================] - 78s 161ms/step - loss: 0.8580 - accuracy: 0.6512 - val_loss: 1.0339 - val_accuracy:
0.6878
Epoch 3/9
480/480 [==============================] - 97s 203ms/step - loss: 0.3734 - accuracy: 0.8744 - val_loss: 0.6565 - val_accuracy:
0.8435
Epoch 4/9
480/480 [==============================] - 79s 165ms/step - loss: 0.2111 - accuracy: 0.9365 - val_loss: 0.6561 - val_accuracy:
0.8234
Epoch 5/9
480/480 [==============================] - 79s 164ms/step - loss: 0.1515 - accuracy: 0.9540 - val_loss: 0.7763 - val_accuracy:
0.8450
Epoch 6/9
480/480 [==============================] - 82s 170ms/step - loss: 0.1204 - accuracy: 0.9633 - val_loss: 0.5966 - val_accuracy:
0.8399
Epoch 7/9
480/480 [==============================] - 67s 139ms/step - loss: 0.1045 - accuracy: 0.9669 - val_loss: 0.7272 - val_accuracy:
0.8440
Epoch 8/9
480/480 [==============================] - 66s 137ms/step - loss: 0.0920 - accuracy: 0.9701 - val_loss: 0.8299 - val_accuracy:
0.8557
Epoch 9/9
480/480 [==============================] - 66s 136ms/step - loss: 0.0846 - accuracy: 0.9724 - val_loss: 0.6389 - val_accuracy:
0.8535

Out[30]: <keras.callbacks.History at 0x20b9cf49190>
```

SAVE THE MODEL

The model is saved using the h5 extension.
It contains multidimensional arrays of scientific data.

```
In [31]: #Saving Model.
         model.save('ECG.h5')
```

TESTING THE MODEL

In [32]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [33]:
```
model=load_model('ECG.h5')
```

In [34]:
```
img=image.load_img("C:/Users/LIKITHA S/Downloads/ibm/Unknown_image.png",target_size=(64,64))
```

In [35]:
```
x=image.img_to_array(img)
```

In [36]:
```
import numpy as np
```

In [37]:
```
x=np.expand_dims(x,axis=0)
```

In [38]:
```
pred = model.predict(x)
y_pred=np.argmax(pred)
y_pred
```

```
1/1 [==============================] - 1s 582ms/step
```

Out[38]:
```
1
```

In [39]:
```
index=['left Bundle Branch block',
       'Normal',
       'Premature Atrial Contraction',
       'Premature Ventricular Contraction',
       'Right Bundle Branch Block',
       'Ventricular Fibrillation']
result = str(index[y_pred])
result
```

Out[39]:
```
'Normal'
```