1. Download the data from here

2. Code the model to classify data like below image

3. Write your own callback function, that has to print the micro F1 score and AUC score after each epoch.

4. Save your model at every epoch if your validation accuracy is improved from previous epoch.

5. you have to decay learning based on below conditions
   Cond1. If your validation accuracy at that epoch is less than previous epoch accuracy, you have to decrese the learning rate by 10%.
   Cond2. For every 3rd epoch, decay your learning rate by 5%.

6. If you are getting any NaN values(either weigths or loss) while training, you have to terminate your training.

7. You have to stop the training if your validation accuracy is not increased in last 2 epochs.

8. Use tensorboard for every model and analyse your gradients. (you need to upload the screenshots for each model for evaluation)

9. use cross entropy as loss function

10. Try the architecture params as given below.

In [111…:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [59]:

```
import tensorflow as tf
import datetime
import numpy as np
import pandas as pd
from warnings import filterwarnings
filterwarnings(action='ignore', module='tensorflow')
filterwarnings(action='ignore', module='tensorboard')
```

In [60]:

```
## Point 1
data = pd.read_csv('/content/drive/MyDrive/Callbacks/data.csv')
print (data.shape)
data.head()
```

```
(20000, 3)
```

Out[60]:

	f1	f2	label
0	0.450564	1.074305	0.0
1	0.085632	0.967682	0.0
2	0.117326	0.971521	1.0
3	0.982179	-0.380408	0.0
4	-0.720352	0.955850	0.0

In [61]:

```
y = data['label'].values
X = data.drop(['label'], axis=1)
```

In [68]:

```
np.unique(y,return_counts=True)
```

Out[68]:

```
(array([0., 1.]), array([10000, 10000]))
```

In [66]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
```

In [70]:

```
print (X_train.shape)
print (X_test.shape)
```

```
(13400, 2)
(6600, 2)
```

In [71]:

```
validation_data=(X_test,y_test)
train_data=(X_train,y_train)
```

Model-1
1. Use tanh as an activation for every layer except output layer.
2. use SGD with momentum as optimizer.
3. use RandomUniform(0,1) as initilizer.
3. Analyze your output and training process.

In [72]:

```
import tensorflow.keras.backend as K
from tensorflow.keras.callbacks import Callback
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Activation
from tensorflow.keras.models import load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import roc_auc_score
```

In [73]:

```
## Point 2
def create_model():
  return tf.keras.models.Sequential([tf.keras.layers.Dense(2, input_shape=(2,), kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, see
                                     tf.keras.layers.Dense(2, activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, see
                                     tf.keras.layers.Dense(2, activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, see
                                     tf.keras.layers.Dense(2, activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, see
                                     tf.keras.layers.Dense(2, activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, see
                                     tf.keras.layers.Dense(1, activation="sigmoid",name='output_layer')])
```

In [74]:

```
## Point 3 - Micro F1 Score
class perf_metrics(Callback):
  def on_train_begin(self, logs={}):
    self.f1score    = []
    self.precisions = []
    self.recalls    = []

  def on_epoch_end(self, epoch, logs={}):
    y_predict   = (np.asarray(self.model.predict(X_test))).round()
    y_actual    = y_test
    val_f1_temp        = f1_score(y_actual,y_predict)
    val_precision_temp = precision_score(y_actual,y_predict)
    val_recall_temp    = recall_score(y_actual,y_predict)
    self.f1score.append(val_f1_temp)
    self.recalls.append(val_recall_temp)
    self.precisions.append(val_precision_temp)
    print ( "F1 Score : %f, Precision : %f, Recall : %f" %(val_f1_temp, val_precision_temp, val_recall_temp))
    return

performance_metrics = perf_metrics()
```

In [75]:

```
## Point 3 AUC
class auc_score(Callback):
  def __init__(self,training_data,validation_data):
    self.x_train = training_data[0]
    self.y_train = training_data[1]
    self.x_test = validation_data[0]
    self.y_test = validation_data[1]

  def on_epoch_end(self, epoch, logs={}):
    y_pred_train = self.model.predict_proba(self.x_train)
    roc_train    = roc_auc_score(self.y_train, y_pred_train)
    y_pred_test  = self.model.predict_proba(self.x_test)
    roc_test     = roc_auc_score(self.y_test, y_pred_test)
    print('ROC_AUC train : %s, ROC_AUC test : %s' % (str(round(roc_train,4)),str(round(roc_test,4)))+'\n')
    return

roc = auc_score(training_data=(X_train, y_train),validation_data=(X_test, y_test))
```

In [76]:

```
## Point 4
filepath        = "/content/drive/MyDrive/Callbacks/model_save/"+"model_1"+"_weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint      = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
```

In [77]:

```
## Point 5-Condition1
reduce_lr_accuracy = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9, patience=1, verbose=1,mode='auto')
```

In [78]:

```
## Point 6
class TerminateNaN(Callback):
  def on_epoch_end(self, epoch, logs={}):
    loss = logs.get('loss')
    if loss is not None:
      if np.isnan(loss) or np.isinf(loss):
        print("Invalid loss and terminated at epoch {}".format(epoch))
        self.model.stop_training = True

terminatenan    = TerminateNaN()
```

In [79]:

```
## Point 7
earlystop = EarlyStopping(monitor='val_accuracy', patience=2, verbose=1,mode='max')
```

In [80]:

```
## Point 8
log_dir ="/content/drive/MyDrive/Callbacks/logs/fit/"+"model_1_" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

In [81]:

```
## Point 5-Condition2

def scheduler(epoch,lr):
  # print (lr)
  # print (type(lr))
  if((epoch+1)%3==0):
    # print (type(lr* 0.95))
    return (lr * 0.95)
  else:
    return lr


lrschedule = LearningRateScheduler(scheduler, verbose=1)
```

In [82]:

```
model_1   = create_model()
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name="SGD")
model_1.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])

callbacks_list  = [performance_metrics,roc,checkpoint,reduce_lr_accuracy,lrschedule,terminatenan,earlystop,tensorboard_callback]
model_1.fit(x=X_train,y=y_train,epochs=20,validation_data=(X_test, y_test),callbacks=[callbacks_list])
```

```
Epoch 1/20

Epoch 00001: LearningRateScheduler reducing learning rate to 0.009999999776482582.
  3/419 [..............................] - ETA: 27s - loss: 0.7776 - accuracy: 0.4792 WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0016s vs `on_train_batch_end` time: 0.0222s). Check your callbacks.
419/419 [==============================] - 2s 3ms/step - loss: 0.6976 - accuracy: 0.5049 - val_loss: 0.6927 - val_accuracy: 0.4974
F1 Score : 0.513137, Precision : 0.497580, Recall : 0.529697
ROC_AUC train : 0.5065, ROC_AUC test : 0.4917


Epoch 00001: val_accuracy improved from -inf to 0.49742, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-01-0.4974.hdf5
Epoch 2/20

Epoch 00002: LearningRateScheduler reducing learning rate to 0.009999999776482582.
419/419 [==============================] - 1s 2ms/step - loss: 0.6912 - accuracy: 0.5194 - val_loss: 0.6920 - val_accuracy: 0.5000
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
F1 Score : 0.000000, Precision : 0.000000, Recall : 0.000000
ROC_AUC train : 0.5066, ROC_AUC test : 0.4917


Epoch 00002: val_accuracy improved from 0.49742 to 0.50000, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-02-0.5000.hdf5
Epoch 3/20

Epoch 00003: LearningRateScheduler reducing learning rate to 0.009499999787658453.
419/419 [==============================] - 1s 2ms/step - loss: 0.6888 - accuracy: 0.5234 - val_loss: 0.6884 - val_accuracy: 0.5135
F1 Score : 0.581192, Precision : 0.510190, Recall : 0.675152
ROC_AUC train : 0.5066, ROC_AUC test : 0.4917


Epoch 00003: val_accuracy improved from 0.50000 to 0.51348, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-03-0.5135.hdf5
Epoch 4/20

Epoch 00004: LearningRateScheduler reducing learning rate to 0.009499999694526196.
419/419 [==============================] - 1s 2ms/step - loss: 0.6855 - accuracy: 0.5354 - val_loss: 0.6853 - val_accuracy: 0.5377
F1 Score : 0.641101, Precision : 0.523938, Recall : 0.825758
ROC_AUC train : 0.5067, ROC_AUC test : 0.4917


Epoch 00004: val_accuracy improved from 0.51348 to 0.53773, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-04-0.5377.hdf5
Epoch 5/20

Epoch 00005: LearningRateScheduler reducing learning rate to 0.009499999694526196.
419/419 [==============================] - 1s 2ms/step - loss: 0.6837 - accuracy: 0.5392 - val_loss: 0.6854 - val_accuracy: 0.5112
F1 Score : 0.575526, Precision : 0.508605, Recall : 0.662727
ROC_AUC train : 0.5648, ROC_AUC test : 0.5584


Epoch 00005: val_accuracy did not improve from 0.53773

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.008549999725073577.
Epoch 6/20

Epoch 00006: LearningRateScheduler reducing learning rate to 0.008122499473392964.
419/419 [==============================] - 1s 2ms/step - loss: 0.6821 - accuracy: 0.5454 - val_loss: 0.6834 - val_accuracy: 0.5395
F1 Score : 0.646010, Precision : 0.524693, Recall : 0.840303
ROC_AUC train : 0.5827, ROC_AUC test : 0.575


Epoch 00006: val_accuracy improved from 0.53773 to 0.53955, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-06-0.5395.hdf5
Epoch 7/20

Epoch 00007: LearningRateScheduler reducing learning rate to 0.008122499100863934.
419/419 [==============================] - 1s 2ms/step - loss: 0.6819 - accuracy: 0.5428 - val_loss: 0.6836 - val_accuracy: 0.5391
F1 Score : 0.648242, Precision : 0.524121, Recall : 0.849394
ROC_AUC train : 0.5946, ROC_AUC test : 0.5858


Epoch 00007: val_accuracy did not improve from 0.53955

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.00731024919077754.
Epoch 8/20

Epoch 00008: LearningRateScheduler reducing learning rate to 0.007310249377042055.
419/419 [==============================] - 1s 2ms/step - loss: 0.6812 - accuracy: 0.5481 - val_loss: 0.6822 - val_accuracy: 0.5409
F1 Score : 0.658937, Precision : 0.524176, Recall : 0.886970
ROC_AUC train : 0.5996, ROC_AUC test : 0.5984


Epoch 00008: val_accuracy improved from 0.53955 to 0.54091, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-08-0.5409.hdf5
Epoch 9/20

Epoch 00009: LearningRateScheduler reducing learning rate to 0.006944736908189952.
419/419 [==============================] - 1s 2ms/step - loss: 0.6808 - accuracy: 0.5484 - val_loss: 0.6816 - val_accuracy: 0.5414
F1 Score : 0.660384, Precision : 0.524319, Recall : 0.891818
ROC_AUC train : 0.6003, ROC_AUC test : 0.598


Epoch 00009: val_accuracy improved from 0.54091 to 0.54136, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-09-0.5414.hdf5
Epoch 10/20

Epoch 00010: LearningRateScheduler reducing learning rate to 0.006944736931473017.
419/419 [==============================] - 1s 2ms/step - loss: 0.6801 - accuracy: 0.5525 - val_loss: 0.6806 - val_accuracy: 0.5418
F1 Score : 0.660225, Precision : 0.524643, Recall : 0.890303
ROC_AUC train : 0.6005, ROC_AUC test : 0.5988


Epoch 00010: val_accuracy improved from 0.54136 to 0.54182, saving model to /content/drive/MyDrive/Callbacks/model_save/model_1_weights-10-0.5418.hdf5
Epoch 11/20

Epoch 00011: LearningRateScheduler reducing learning rate to 0.006944736931473017.
419/419 [==============================] - 1s 2ms/step - loss: 0.6795 - accuracy: 0.5516 - val_loss: 0.6829 - val_accuracy: 0.5397
F1 Score : 0.653592, Precision : 0.523949, Recall : 0.868485
ROC_AUC train : 0.6, ROC_AUC test : 0.5979


Epoch 00011: val_accuracy did not improve from 0.54182

Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.006250263238325715.
Epoch 12/20

Epoch 00012: LearningRateScheduler reducing learning rate to 0.005937750032171607.
419/419 [==============================] - 1s 2ms/step - loss: 0.6789 - accuracy: 0.5576 - val_loss: 0.6800 - val_accuracy: 0.5398
F1 Score : 0.656720, Precision : 0.523707, Recall : 0.880303
ROC_AUC train : 0.6018, ROC_AUC test : 0.5999


Epoch 00012: val_accuracy did not improve from 0.54182

Epoch 00012: ReduceLROnPlateau reducing learning rate to 0.005343974987044931.
Epoch 00012: early stopping
```

Out[82]:

```
<tensorflow.python.keras.callbacks.History at 0x7f3862335090>
```

In [83]:

```
model_1.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 2)                 6
_________________________________________________________________
layer2 (Dense)               (None, 2)                 6
_________________________________________________________________
layer3 (Dense)               (None, 2)                 6
_________________________________________________________________
layer4 (Dense)               (None, 2)                 6
_________________________________________________________________
layer5 (Dense)               (None, 2)                 6
_________________________________________________________________
output_layer (Dense)         (None, 1)                 3
=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
```

In [84]:

```
%reload_ext tensorboard
!rm -rf ./logs/
%tensorboard --logdir='/content/drive/MyDrive/Callbacks/logs/fit/'
```

```
Output hidden; open in https://colab.research.google.com to view.
```

Model-2
1. Use relu as an activation for every layer except output layer.
2. use SGD with momentum as optimizer.
3. use RandomUniform(0,1) as initilizer.
3. Analyze your output and training process.

In [86]:

```
def create_model2():
  return tf.keras.models.Sequential([tf.keras.layers.Dense(2, input_shape=(2,),kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, seed
                                     tf.keras.layers.Dense(2, activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, seed
                                     tf.keras.layers.Dense(2, activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, seed
                                     tf.keras.layers.Dense(2, activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, seed
                                     tf.keras.layers.Dense( 2,activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(minval=0, maxval=1, seed
                                     tf.keras.layers.Dense(1,activation='sigmoid',name='output_layer')])
```

In [88]:

```
filepath        = "/content/drive/MyDrive/Callbacks/model_save/"+"model_2"+"_weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint      = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

log_dir         ="/content/drive/MyDrive/Callbacks/logs/fit/"+"model_2_" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

In [89]:

```
model_2   = create_model2()
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name="SGD")
model_2.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])

callbacks_list  = [checkpoint,tensorboard_callback,earlystop,reduce_lr_accuracy,lrschedule,terminatenan,performance_metrics,roc]
model_2.fit(x=X_train,y=y_train,epochs=20,validation_data=(X_test, y_test),callbacks=[callbacks_list])
```

```
Epoch 1/20

Epoch 00001: LearningRateScheduler reducing learning rate to 0.009999999776482582.
  3/419 [..............................] - ETA: 26s - loss: 0.6647 - accuracy: 0.5729 WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0024s vs `on_train_batch_end` time: 0.0213s). Check your callbacks.
419/419 [==============================] - 2s 3ms/step - loss: 0.6851 - accuracy: 0.5366 - val_loss: 0.6841 - val_accuracy: 0.5338

Epoch 00001: val_accuracy improved from -inf to 0.53379, saving model to /content/drive/MyDrive/Callbacks/model_save/model_2_weights-01-0.5338.hdf5
F1 Score : 0.630834, Precision : 0.522145, Recall : 0.796667
ROC_AUC train : 0.5515, ROC_AUC test : 0.5388

Epoch 2/20

Epoch 00002: LearningRateScheduler reducing learning rate to 0.009999999776482582.
419/419 [==============================] - 1s 2ms/step - loss: 0.6822 - accuracy: 0.5453 - val_loss: 0.6826 - val_accuracy: 0.5424

Epoch 00002: val_accuracy improved from 0.53379 to 0.54242, saving model to /content/drive/MyDrive/Callbacks/model_save/model_2_weights-02-0.5424.hdf5
F1 Score : 0.661511, Precision : 0.524902, Recall : 0.894242
ROC_AUC train : 0.5525, ROC_AUC test : 0.5441

Epoch 3/20

Epoch 00003: LearningRateScheduler reducing learning rate to 0.009499999787658453.
419/419 [==============================] - 1s 2ms/step - loss: 0.6824 - accuracy: 0.5451 - val_loss: 0.6816 - val_accuracy: 0.5427

Epoch 00003: val_accuracy improved from 0.54242 to 0.54273, saving model to /content/drive/MyDrive/Callbacks/model_save/model_2_weights-03-0.5427.hdf5
F1 Score : 0.657279, Precision : 0.525608, Recall : 0.876970
ROC_AUC train : 0.5521, ROC_AUC test : 0.543

Epoch 4/20

Epoch 00004: LearningRateScheduler reducing learning rate to 0.009499999694526196.
419/419 [==============================] - 1s 2ms/step - loss: 0.6815 - accuracy: 0.5482 - val_loss: 0.6828 - val_accuracy: 0.5392

Epoch 00004: val_accuracy did not improve from 0.54273

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.008549999725073577.
F1 Score : 0.650017, Precision : 0.524030, Recall : 0.855758
ROC_AUC train : 0.5506, ROC_AUC test : 0.5422

Epoch 5/20

Epoch 00005: LearningRateScheduler reducing learning rate to 0.008549999445676804.
419/419 [==============================] - 1s 2ms/step - loss: 0.6815 - accuracy: 0.5493 - val_loss: 0.6854 - val_accuracy: 0.5380

Epoch 00005: val_accuracy did not improve from 0.54273

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.007694999501109123.
F1 Score : 0.642010, Precision : 0.524056, Recall : 0.828485
ROC_AUC train : 0.5507, ROC_AUC test : 0.5392

Epoch 00005: early stopping
```

Out[89]:

```
<tensorflow.python.keras.callbacks.History at 0x7f38620e0990>
```

In [90]:

```
model_2.summary()
```

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 2)                 6
_________________________________________________________________
layer2 (Dense)               (None, 2)                 6
_________________________________________________________________
layer3 (Dense)               (None, 2)                 6
_________________________________________________________________
layer4 (Dense)               (None, 2)                 6
_________________________________________________________________
layer5 (Dense)               (None, 2)                 6
_________________________________________________________________
output_layer (Dense)         (None, 1)                 3
=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
```

In [91]:

```
%reload_ext tensorboard
!rm -rf ./logs/
%tensorboard --logdir='/content/drive/MyDrive/Callbacks/logs/fit/'
```

```
Output hidden; open in https://colab.research.google.com to view.
```

Model-3
1. Use relu as an activation for every layer except output layer.
2. use SGD with momentum as optimizer.
3. use he_uniform() as initilizer.
3. Analyze your output and training process.

In [92]:

```
def create_model3():
  return tf.keras.models.Sequential([tf.keras.layers.Dense(2, input_shape=(2,),kernel_initializer=tf.keras.initializers.HeUniform(),name='layer1'),
tf.keras.layers.Dense(2, activation='relu',kernel_initializer=tf.keras.initializers.HeUniform(),name='layer2'),
tf.keras.layers.Dense(2, activation='relu',kernel_initializer=tf.keras.initializers.HeUniform(),name='layer3'),
tf.keras.layers.Dense(2, activation='relu',kernel_initializer=tf.keras.initializers.HeUniform(),name='layer4'),
tf.keras.layers.Dense( 2,activation='relu',kernel_initializer=tf.keras.initializers.HeUniform(),name='layer5'),
tf.keras.layers.Dense(1,activation="sigmoid",name='output_layer')])
```

In [93]:

```
filepath        = "/content/drive/MyDrive/Callbacks/model_save/"+"model_3"+"_weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint      = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

log_dir         ="/content/drive/MyDrive/Callbacks/logs/fit/"+"model_3_" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

In [95]:

```
model_3 = create_model3()
optimizer     = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name="SGD")
model_3.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])

callbacks_list  = [checkpoint,tensorboard_callback,earlystop,reduce_lr_accuracy,lrschedule,terminatenan,performance_metrics,roc]
model_3.fit(x=X_train,y=y_train,epochs=20,validation_data=(X_test, y_test),callbacks=[callbacks_list])
```

```
Epoch 1/20

Epoch 00001: LearningRateScheduler reducing learning rate to 0.009999999776482582.
  3/419 [..............................] - ETA: 20s - loss: 0.8054 - accuracy: 0.4479 WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0018s vs `on_train_batch_end` time: 0.0158s). Check your callbacks.
419/419 [==============================] - 1s 2ms/step - loss: 0.6850 - accuracy: 0.5451 - val_loss: 0.6854 - val_accuracy: 0.5324

Epoch 00001: val_accuracy improved from 0.50000 to 0.53242, saving model to /content/drive/MyDrive/Callbacks/model_save/model_3_weights-01-0.5324.hdf5
F1 Score : 0.626392, Precision : 0.521573, Recall : 0.783939
ROC_AUC train : 0.5439, ROC_AUC test : 0.535

Epoch 2/20

Epoch 00002: LearningRateScheduler reducing learning rate to 0.009999999776482582.
419/419 [==============================] - 1s 2ms/step - loss: 0.6789 - accuracy: 0.5552 - val_loss: 0.6722 - val_accuracy: 0.5791

Epoch 00002: val_accuracy improved from 0.53242 to 0.57909, saving model to /content/drive/MyDrive/Callbacks/model_save/model_3_weights-02-0.5791.hdf5
F1 Score : 0.629006, Precision : 0.562321, Recall : 0.713636
ROC_AUC train : 0.6026, ROC_AUC test : 0.6002

Epoch 3/20

Epoch 00003: LearningRateScheduler reducing learning rate to 0.009499999787658453.
419/419 [==============================] - 1s 2ms/step - loss: 0.6701 - accuracy: 0.5787 - val_loss: 0.6748 - val_accuracy: 0.5698

Epoch 00003: val_accuracy did not improve from 0.57909

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.008549999725073577.
F1 Score : 0.652807, Precision : 0.547263, Recall : 0.808788
ROC_AUC train : 0.5915, ROC_AUC test : 0.5842

Epoch 4/20

Epoch 00004: LearningRateScheduler reducing learning rate to 0.008549999445676804.
419/419 [==============================] - 1s 2ms/step - loss: 0.6696 - accuracy: 0.5821 - val_loss: 0.6681 - val_accuracy: 0.5894

Epoch 00004: val_accuracy improved from 0.57909 to 0.58939, saving model to /content/drive/MyDrive/Callbacks/model_save/model_3_weights-04-0.5894.hdf5
F1 Score : 0.665432, Precision : 0.561458, Recall : 0.816667
ROC_AUC train : 0.6047, ROC_AUC test : 0.6067

Epoch 5/20

Epoch 00005: LearningRateScheduler reducing learning rate to 0.008549999445676804.
419/419 [==============================] - 1s 2ms/step - loss: 0.6684 - accuracy: 0.5878 - val_loss: 0.6689 - val_accuracy: 0.5864

Epoch 00005: val_accuracy did not improve from 0.58939

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.007694999501109123.
F1 Score : 0.643603, Precision : 0.565367, Recall : 0.746970
ROC_AUC train : 0.6068, ROC_AUC test : 0.6028

Epoch 6/20

Epoch 00006: LearningRateScheduler reducing learning rate to 0.007310249703004956.
419/419 [==============================] - 1s 2ms/step - loss: 0.6677 - accuracy: 0.5881 - val_loss: 0.6678 - val_accuracy: 0.5867

Epoch 00006: val_accuracy did not improve from 0.58939

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.0065792248584330085.
F1 Score : 0.632246, Precision : 0.569451, Recall : 0.710606
ROC_AUC train : 0.6086, ROC_AUC test : 0.604

Epoch 00006: early stopping
```

Out[95]:

```
<tensorflow.python.keras.callbacks.History at 0x7f385d3a9390>
```

In [96]:

```
model_3.summary()
```

```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 2)                 6
_________________________________________________________________
layer2 (Dense)               (None, 2)                 6
_________________________________________________________________
layer3 (Dense)               (None, 2)                 6
_________________________________________________________________
layer4 (Dense)               (None, 2)                 6
_________________________________________________________________
layer5 (Dense)               (None, 2)                 6
_________________________________________________________________
output_layer (Dense)         (None, 1)                 3
=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
```

In [97]:

```
%reload_ext tensorboard
!rm -rf ./logs/
%tensorboard --logdir='/content/drive/MyDrive/Callbacks/logs/fit/'
```

```
Output hidden; open in https://colab.research.google.com to view.
```

Model-4
1. Try with any values to get better accuracy/f1 score.

In [106…:

```
def create_model4():
  return tf.keras.models.Sequential([tf.keras.layers.Dense(2, input_shape=(2,),kernel_initializer=tf.keras.initializers.GlorotNormal(seed=None),name='layer1'),
tf.keras.layers.Dense(2, activation='tanh',kernel_initializer=tf.keras.initializers.GlorotNormal(seed=None),name='layer2'),
tf.keras.layers.Dense(2, activation='tanh',kernel_initializer=tf.keras.initializers.GlorotNormal(seed=None),name='layer3'),
tf.keras.layers.Dense(2, activation='tanh',kernel_initializer=tf.keras.initializers.GlorotNormal(seed=None),name='layer4'),
tf.keras.layers.Dense(2,activation='tanh',kernel_initializer=tf.keras.initializers.GlorotNormal(seed=None),name='layer5'),
tf.keras.layers.Dense(1,activation="sigmoid",name='output_layer')])
```

In [107…:

```
filepath        = "/content/drive/MyDrive/Callbacks/model_save/"+"model_4"+"_weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint      = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

log_dir         ="/content/drive/MyDrive/Callbacks/logs/fit/"+"model_4_" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

In [108…:

```
model_4 = create_model4()
optimizer     = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name="SGD")
model_4.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])

callbacks_list  = [checkpoint,tensorboard_callback,earlystop,reduce_lr_accuracy,lrschedule,terminatenan,performance_metrics,roc]
model_4.fit(x=X_train,y=y_train,epochs=20,validation_data=(X_test, y_test),callbacks=[callbacks_list])
```

```
Epoch 1/20

Epoch 00001: LearningRateScheduler reducing learning rate to 0.009999999776482582.
  3/419 [..............................] - ETA: 27s - loss: 0.6908 - accuracy: 0.5521 WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0023s vs `on_train_batch_end` time: 0.0215s). Check your callbacks.
419/419 [==============================] - 2s 3ms/step - loss: 0.6938 - accuracy: 0.5078 - val_loss: 0.6960 - val_accuracy: 0.5000

Epoch 00001: val_accuracy improved from -inf to 0.50000, saving model to /content/drive/MyDrive/Callbacks/model_save/model_4_weights-01-0.5000.hdf5
F1 Score : 0.666667, Precision : 0.500000, Recall : 1.000000
ROC_AUC train : 0.5309, ROC_AUC test : 0.5213

Epoch 2/20

Epoch 00002: LearningRateScheduler reducing learning rate to 0.009999999776482582.
419/419 [==============================] - 1s 2ms/step - loss: 0.6915 - accuracy: 0.5243 - val_loss: 0.6843 - val_accuracy: 0.5744

Epoch 00002: val_accuracy improved from 0.50000 to 0.57439, saving model to /content/drive/MyDrive/Callbacks/model_save/model_4_weights-02-0.5744.hdf5
F1 Score : 0.592840, Precision : 0.568213, Recall : 0.619697
ROC_AUC train : 0.5958, ROC_AUC test : 0.5827

Epoch 3/20

Epoch 00003: LearningRateScheduler reducing learning rate to 0.009499999787658453.
419/419 [==============================] - 1s 2ms/step - loss: 0.6724 - accuracy: 0.5954 - val_loss: 0.6652 - val_accuracy: 0.6026

Epoch 00003: val_accuracy improved from 0.57439 to 0.60258, saving model to /content/drive/MyDrive/Callbacks/model_save/model_4_weights-03-0.6026.hdf5
F1 Score : 0.656900, Precision : 0.577906, Recall : 0.760909
ROC_AUC train : 0.6092, ROC_AUC test : 0.6022

Epoch 4/20

Epoch 00004: LearningRateScheduler reducing learning rate to 0.009499999694526196.
419/419 [==============================] - 1s 2ms/step - loss: 0.6665 - accuracy: 0.5971 - val_loss: 0.6651 - val_accuracy: 0.5976

Epoch 00004: val_accuracy did not improve from 0.60258

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.008549999725073577.
F1 Score : 0.670635, Precision : 0.567590, Recall : 0.819394
ROC_AUC train : 0.605, ROC_AUC test : 0.5991

Epoch 5/20

Epoch 00005: LearningRateScheduler reducing learning rate to 0.008549999445676804.
419/419 [==============================] - 1s 2ms/step - loss: 0.6636 - accuracy: 0.6022 - val_loss: 0.6610 - val_accuracy: 0.6023

Epoch 00005: val_accuracy did not improve from 0.60258

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.007694999501109123.
F1 Score : 0.659665, Precision : 0.576479, Recall : 0.770909
ROC_AUC train : 0.6115, ROC_AUC test : 0.6028

Epoch 00005: early stopping
```

Out[108…:

```
<tensorflow.python.keras.callbacks.History at 0x7f3853d53390>
```

In [109…:

```
model_4.summary()
```

```
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 2)                 6
_________________________________________________________________
layer2 (Dense)               (None, 2)                 6
_________________________________________________________________
layer3 (Dense)               (None, 2)                 6
_________________________________________________________________
layer4 (Dense)               (None, 2)                 6
_________________________________________________________________
layer5 (Dense)               (None, 2)                 6
_________________________________________________________________
output_layer (Dense)         (None, 1)                 3
=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
```

In [110…:

```
%reload_ext tensorboard
!rm -rf ./logs/
%tensorboard --logdir='/content/drive/MyDrive/Callbacks/logs/fit/'
```

```
Output hidden; open in https://colab.research.google.com to view.
```

In [ ]: