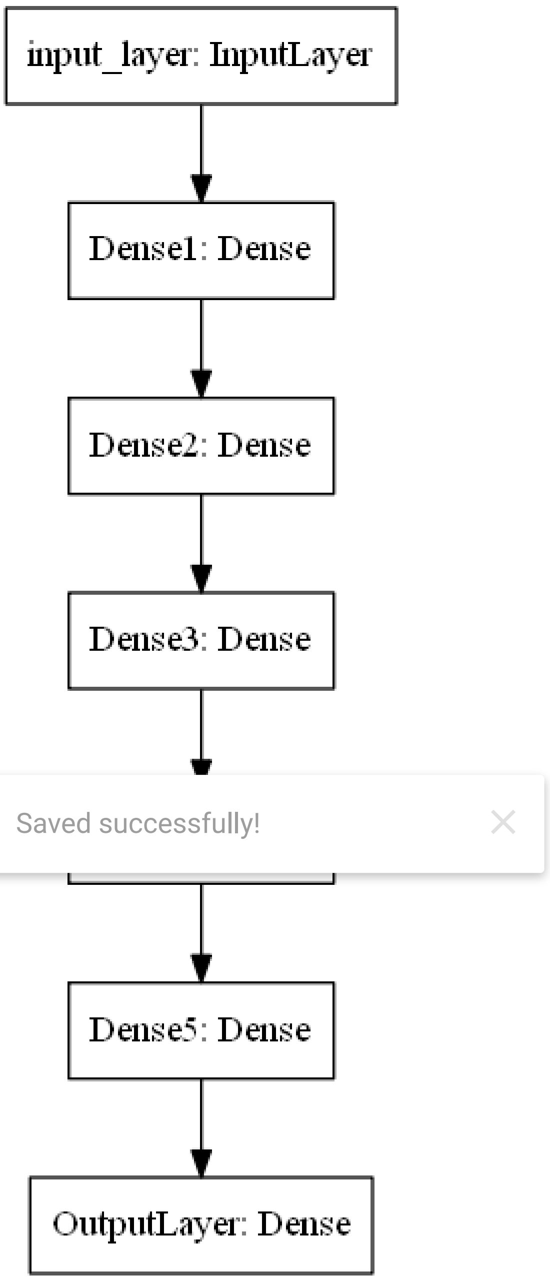
1. Download the data from here. You have to use data.csv file for this assignment

2. Code the model to classify data like below image. You can use any number of units in your Dense layers.



3. Writing Callbacks

You have to implement the following callbacks

- Write your own callback function, that has to print the micro F1 score and AUC score after each epoch.Do not use tf.keras.metrics for calculating AUC and F1 score.

- Save your model at every epoch if your validation accuracy is improved from previous epoch.
- You have to decay learning based on below conditions

```
Cond1. If your validation accuracy at that epoch is less than previous epoch a
       learning rate by 10%.
Cond2. For every 3rd epoch, decay your learning rate by 5%.
```

- If you are getting any NaN values(either weigths or loss) while training, you have to terminate your training.
- You have to stop the training if your validation accuracy is not increased in last 2 epochs.
- Use tensorboard for every model and analyse your scalar plots and histograms. (you need to upload the screenshots and write the observations for each model for evaluation)

`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.
```

Saved successfully!

Writing Callbacks

Write your own callback function, that has to print the micro F1 score and AUC score after each epoch.Do not use tf.keras.metrics for calculating AUC and F1 score.

```
from google.colab import files
files=files.upload()
```

Choose Files data.csv

- **data.csv**(application/vnd.ms-excel) - 886913 bytes, last modified: 12/27/2021 - 100% done

```
Saving data.csv to data (1).csv
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
import random as rn
import tensorflow as tf
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
from keras.callbacks import ReduceLROnPlateau
```

```
data=pd.read_csv("data (1).csv")
data.head()
```

	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

Saved successfully!

```
ues
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, stratify=Y)
#X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratif
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)

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

```
class Metrics(tf.keras.callbacks.Callback):

   def on_train_begin(self, logs={}):
     self.val_f1s = []

   def on_epoch_end(self, epoch, logs={}):
     #val_predict = (np.asarray(self.model.predict(self.model.validation_data[0]))).round(
     val_predict = (np.asarray(self.model.predict(X_test))).round()
```

```
        #val_targ = self.model.validation_data[1]
        _val_f1 = f1_score(y_test, val_predict,average='micro')
        self.val_f1s.append(_val_f1)
        #print(" value f1 ",_val_f1)
        print("  f1_score: "+"{:.4f}".format(_val_f1));
        return

history_own=Metrics()
#print(history_own.val_f1s)


#ROC_ AUC Score:

from sklearn.metrics import roc_auc_score
from keras.callbacks import Callback

class RocCallback(Callback):
    def __init__(self,training_data,validation_data):
        self.x = training_data[0]
        self.y = training_data[1]
        self.x_val = validation_data[0]
        self.y_val = validation_data[1]



    def on_train_begin(self, logs={}):
        return

    def on_train_end(self, logs={}):
        return

    def on_epoch_begin(self, epoch, logs={}):
```

Saved successfully!

```
    def on_epoch_end(self, epoch, logs={}):
        y_pred_train = self.model.predict_on_batch(self.x) #model.predict_on_batch(X_test)
        roc_train = roc_auc_score(self.y, y_pred_train)
        y_pred_val = self.model.predict_on_batch(self.x_val)
        roc_val = roc_auc_score(self.y_val, y_pred_val)
        print('\rroc-auc_train: %s - roc-auc_val: %s' % (str(round(roc_train,4)),str(round
        return

    def on_batch_begin(self, batch, logs={}):
        return

    def on_batch_end(self, batch, logs={}):
        return

roc = RocCallback(training_data=(X_train, y_train),
                  validation_data=(X_test, y_test))

#model.fit(X_train, y_train, validation_data=(X_test, y_test),callbacks=[roc])


#Input layer
input_layer = Input(shape=(2,))
```

```
#Dense hidden layer
layer1 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(

#output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.RandomUnifo

#Model Creation
model = Model(inputs=input_layer,outputs=output)

#Now Callbacks:
#history_own = LossHistory()
history_own  = Metrics()

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.09, nesterov=False, nam

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.0001, v

model.compile(optimizer=optimizer, loss='BinaryCrossentropy')

model.fit(X_train,y_train,validation_split=0.3,shuffle=True,verbose=1, epochs=5, validatio
```

```
Epoch 1/5
461/469 [============================>.] - ETA: 0s - loss: 0.7255  f1_score: 0.5032
roc-auc_train: 0.5019 - roc-auc_val: 0.4955
469/469 [==============================] - 2s 3ms/step - loss: 0.7251 - val_loss: 0.7
Epoch 2/5
430/469 [==========================>...] - ETA: 0s - loss: 0.6957  f1_score: 0.4982
roc-auc_train: 0.5053 - roc-auc_val: 0.4988
469/469 [==============================] - 1s 2ms/step - loss: 0.6953 - val_loss: 0.6

  Saved successfully!                 ======>.] - ETA: 0s - loss: 0.6931  f1_score: 0.5126
roc-auc_train: 0.5122 - roc-auc_val: 0.507
469/469 [==============================] - 1s 3ms/step - loss: 0.6931 - val_loss: 0.6
Epoch 4/5
465/469 [============================>.] - ETA: 0s - loss: 0.6926  f1_score: 0.5164
roc-auc_train: 0.5187 - roc-auc_val: 0.5121
469/469 [==============================] - 1s 2ms/step - loss: 0.6926 - val_loss: 0.6
Epoch 5/5
433/469 [==========================>...] - ETA: 0s - loss: 0.6923  f1_score: 0.5203
roc-auc_train: 0.5273 - roc-auc_val: 0.521
469/469 [==============================] - 1s 2ms/step - loss: 0.6923 - val_loss: 0.6
<keras.callbacks.History at 0x7f79f25e6990>
```

```
history_own.val_f1s
```

```
[0.5031818181818182,
 0.49818181818181817,
 0.5125757575757576,
 0.5163636363636364,
 0.5203030303030303]
```

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

```
class TerminateNaN(tf.keras.callbacks.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

    def epoch_end(self,epoch):
      model_weights = self.model.get_weights()
      if model_weights is not None:
        if np.any([np.any(np.isnan(x)) for x in model_weights]):
          print("Invalid weights and terminated at epoch{}".format(epoch))

          self.model.stop_training = True


terminate= TerminateNaN()
model.fit(X_train,y_train,epochs=5,validation_data=(X_test,y_test),batch_size=20,callbacks
```

```
     Epoch 1/5
     670/670 [==============================] - 1s 2ms/step - loss: 0.6921 - val_loss: 0.6
     Epoch 2/5
     670/670 [==============================] - 1s 2ms/step - loss: 0.6916 - val_loss: 0.6
                                      ========] - 1s 2ms/step - loss: 0.6910 - val_loss: 0.6

     670/670 [==============================] - 1s 2ms/step - loss: 0.6905 - val_loss: 0.6
     Epoch 5/5
     670/670 [==============================] - 1s 2ms/step - loss: 0.6898 - val_loss: 0.6
     <keras.callbacks.History at 0x7f79e9020950>
```

Saved successfully! ×

```
#Save your model at every epoch if your validation accuracy is improved from previous epoc

#Input layer
input_layer = Input(shape=(2,))
#Dense hidden layer
layer1 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(
#output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.RandomUnifo
#Creating a model
model = Model(inputs=input_layer,outputs=output)

#Callbacks
#file path, it saves the model in the 'model_save' folder and we are naming model with epo
#and val auc to differtiate with other models
#you have to create model_save folder before running the code.

filepath="D:\Applied AI Course\Assignments\20. Assignment- Working with Callbacks\model_sa
```

```
filepath= D:\Applied AI Course\Assignments\20. Assignment- Working with Callbacks\model_sa
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_o

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.09, nesterov=False, nam

auc=tf.keras.metrics.AUC()

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.0001, v

model.compile(optimizer=optimizer, loss='BinaryCrossentropy')

model.fit(X_train,y_train,epochs=5,validation_data=(X_test,y_test),batch_size=20,callbacks
```

```
Epoch 1/5
662/670 [============================>.] - ETA: 0s - loss: 0.7382WARNING:tensorflow:(
roc-auc_train: 0.5027 - roc-auc_val: 0.4957
670/670 [==============================] - 2s 2ms/step - loss: 0.7375 - val_loss: 0.6
Epoch 2/5
659/670 [============================>.] - ETA: 0s - loss: 0.6937WARNING:tensorflow:(
roc-auc_train: 0.5176 - roc-auc_val: 0.5053
670/670 [==============================] - 1s 2ms/step - loss: 0.6937 - val_loss: 0.6
Epoch 3/5
644/670 [===========================>..] - ETA: 0s - loss: 0.6926WARNING:tensorflow:(
roc-auc_train: 0.5526 - roc-auc_val: 0.5461
670/670 [==============================] - 2s 2ms/step - loss: 0.6926 - val_loss: 0.6
Epoch 4/5
650/670 [============================>.] - ETA: 0s - loss: 0.6923WARNING:tensorflow:(
roc-auc_train: 0.5675 - roc-auc_val: 0.5609
670/670 [==============================] - 2s 2ms/step - loss: 0.6923 - val_loss: 0.6
Epoch 5/5
637/670 [===========================>..] - ETA: 0s - loss: 0.6919WARNING:tensorflow:(
roc-auc_train: 0.5883 - roc-auc_val: 0.5739
                        ========] - 2s 2ms/step - loss: 0.6919 - val_loss: 0.6
                        f79e91fbd50>
```

Saved successfully!

```
#You have to stop the training if your validation accuracy is not increased in last 2 epoc

#Input layer
input_layer = Input(shape=(2,))
#Dense hidden layer
layer1 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(
#output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.RandomUnifo
#Creating a model
model = Model(inputs=input_layer,outputs=output)

earlystop = EarlyStopping(monitor='val_auc', patience=2, verbose=1, mode='max')

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.09, nesterov=False, nam

auc=tf.keras.metrics.AUC()

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.0001, v

model.compile(optimizer=optimizer, loss='BinaryCrossentropy')
model.fit(X_train,y_train,epochs=5,validation_data=(X_test,y_test),batch_size=20,callbacks
```

```
Epoch 1/5
643/670 [===========================>..] - ETA: 0s - loss: 0.7550WARNING:tensorflow:E
roc-auc_train: 0.5012 - roc-auc_val: 0.4956
670/670 [==============================] - 3s 4ms/step - loss: 0.7527 - val_loss: 0.6
Epoch 2/5
648/670 [============================>.] - ETA: 0s - loss: 0.6941WARNING:tensorflow:E
roc-auc_train: 0.5074 - roc-auc_val: 0.5015
670/670 [==============================] - 2s 3ms/step - loss: 0.6941 - val_loss: 0.6
Epoch 3/5
654/670 [============================>.] - ETA: 0s - loss: 0.6931WARNING:tensorflow:E
roc-auc_train: 0.548 - roc-auc_val: 0.5493
670/670 [==============================] - 2s 3ms/step - loss: 0.6931 - val_loss: 0.6
Epoch 4/5
667/670 [============================>.] - ETA: 0s - loss: 0.6929WARNING:tensorflow:E
roc-auc_train: 0.5558 - roc-auc_val: 0.5579
670/670 [==============================] - 2s 3ms/step - loss: 0.6929 - val_loss: 0.6
Epoch 5/5
651/670 [============================>.] - ETA: 0s - loss: 0.6927WARNING:tensorflow:E
roc-auc_train: 0.5673 - roc-auc_val: 0.5709
670/670 [==============================] - 2s 4ms/step - loss: 0.6927 - val_loss: 0.6
<keras.callbacks.History at 0x7f79f250c4d0>
```

```
#You have to decay learning rate on the basis of following conditions:

#Cond1. If your validation accuracy at that epoch is less than previous epoch accuracy, yo
#Cond2. For every 3rd epoch, decay your learning rate by 5%.
```

Saved successfully!

```
#    if ((epoch+1) % 3 ==0):
#      changed = initial_learningrate*(1-0.05)**(epoch+1)
#    else:
#      changed = initial_learningrate*(1-0.1)**(epoch+1)
#    return changed

#changed_lr = []
#for i in range(1,10):
#  changed_lr.append(changeLearningRate(i))

from tensorflow.keras.callbacks import LearningRateScheduler

def scheduler(epoch,lr):

  if((epoch+1)%3==0):
    lr=0.95*lr
  return lr

#changed_lr = []
#for i in range(1,10):
#  changed_lr.append(scheduler(i))

#Input layer
```

```
input_layer = Input(shape=(2,))
#Dense hidden layer
layer1 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(
#output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.RandomUnifo
#Creating a model
model = Model(inputs=input_layer,outputs=output)

lrschedule = LearningRateScheduler(scheduler, verbose=0.1)

filepath="D:\Applied AI Course\Assignments\20. Assignment- Working with Callbacks\model_sa
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_o
earlystop = EarlyStopping(monitor='val_auc', patience=2, verbose=1, mode='max')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.0001, v

# here we are creating a list with all the callbacks we want
callback_list = [reduce_lr,history_own,lrschedule, earlystop, checkpoint,terminate]

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.09, nesterov=False, nam

model.compile(optimizer=optimizer, loss='BinaryCrossentropy')

model.fit(X_train,y_train,epochs=20,validation_data=(X_test,y_test),batch_size=20,callback
```

```
Epoch 00001: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/20
645/670 [===========================>..] - ETA: 0s - loss: 0.7175  f1_score: 0.498
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.5005 - roc-auc_val: 0.4959
Saved successfully!           =========] - 3s 4ms/step - loss: 0.7167 - val_loss: (

Epoch 00002: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/20
661/670 [============================>.] - ETA: 0s - loss: 0.6945  f1_score: 0.497
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.5038 - roc-auc_val: 0.4996
670/670 [==============================] - 3s 4ms/step - loss: 0.6944 - val_loss: (

Epoch 00003: LearningRateScheduler setting learning rate to 0.009499999787658453.
Epoch 3/20
652/670 [============================>.] - ETA: 0s - loss: 0.6929  f1_score: 0.537
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.5401 - roc-auc_val: 0.5338
670/670 [==============================] - 2s 3ms/step - loss: 0.6929 - val_loss: (

Epoch 00004: LearningRateScheduler setting learning rate to 0.009499999694526196.
Epoch 4/20
646/670 [===========================>..] - ETA: 0s - loss: 0.6926  f1_score: 0.538
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.5466 - roc-auc_val: 0.5357
670/670 [==============================] - 1s 2ms/step - loss: 0.6926 - val_loss: (

Epoch 00005: LearningRateScheduler setting learning rate to 0.009499999694526196.
Epoch 5/20
667/670 [============================>.] - ETA: 0s - loss: 0.6924  f1_score: 0.579
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6109 - roc-auc_val: 0.6011
670/670 [==============================] - 2s 2ms/step - loss: 0.6924 - val_loss: 

Epoch 00006: LearningRateScheduler setting learning rate to 0.009024999709799886.
Epoch 6/20
653/670 [============================>.] - ETA: 0s - loss: 0.6921  f1_score: 0.560
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6133 - roc-auc_val: 0.5978
670/670 [==============================] - 2s 2ms/step - loss: 0.6921 - val_loss: 

Epoch 00007: LearningRateScheduler setting learning rate to 0.009025000035762787.
Epoch 7/20
670/670 [==============================] - ETA: 0s - loss: 0.6918  f1_score: 0.571
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6061 - roc-auc_val: 0.5907
670/670 [==============================] - 2s 2ms/step - loss: 0.6918 - val_loss: 
```

Model 1 Observations:

1. Epoch No. 8 given Maximum F1 Score: 0.5900 & roc-auc_val: 0.6065
2. As Epoch number increases, val_loss decreases

Saved successfully!

```
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.
```

```
#Input layer
input_layer = Input(shape=(2,))
#Dense hidden layer
layer1 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(
#output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.RandomUnifo
#Creating a model
model = Model(inputs=input_layer,outputs=output)

lrschedule = LearningRateScheduler(scheduler, verbose=0.1)
filepath="D:\Applied AI Course\Assignments\20. Assignment- Working with Callbacks\model_sa
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_o
earlystop = EarlyStopping(monitor='val_auc', patience=2, verbose=1, mode='max')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.0001, v


# here we are creating a list with all the callbacks we want
callback_list = [history_own,lrschedule, earlystop, checkpoint,terminate]

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.09, nesterov=False, nam

model.compile(optimizer=optimizer, loss='BinaryCrossentropy')

model.fit(X_train,y_train,epochs=20,validation_data=(X_test,y_test),batch_size=20,callback
```

```
    roc-auc_train: 0.5309 - roc-auc_val: 0.5289
    670/670 [==============================] - 2s 2ms/step - loss: 0.6892 - val_loss:

    Epoch 00008: LearningRateScheduler setting learning rate to 0.009025000035762787.
    Epoch 8/20
    637/670 [===========================>..] - ETA: 0s - loss: 0.6888  f1_score: 0.519
    WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
    WARNING:tensorflow:Can save best model only with val_auc available, skipping.
    roc-auc_train: 0.5309 - roc-auc_val: 0.5299
    670/670 [==============================] - 2s 2ms/step - loss: 0.6889 - val_loss:

    Epoch 00009: LearningRateScheduler setting learning rate to 0.008573750033974648.
    Epoch 9/20
    646/670 [===========================>..] - ETA: 0s - loss: 0.6886  f1_score: 0.521
    WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
    WARNING:tensorflow:Can save best model only with val_auc available, skipping.
    roc-auc_train: 0.5308 - roc-auc_val: 0.5294
    670/670 [==============================] - 2s 2ms/step - loss: 0.6887 - val_loss:

    Saved successfully!                    ler setting learning rate to 0.008573750033974648.

    635/670 [===========================>..] - ETA: 0s - loss: 0.6884  f1_score: 0.525
    WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
    WARNING:tensorflow:Can save best model only with val_auc available, skipping.
    roc-auc_train: 0.531 - roc-auc_val: 0.5287
    670/670 [==============================] - 2s 3ms/step - loss: 0.6884 - val_loss:

    Epoch 00011: LearningRateScheduler setting learning rate to 0.008573750033974648.
    Epoch 11/20
    650/670 [============================>.] - ETA: 0s - loss: 0.6879  f1_score: 0.526
    WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
    WARNING:tensorflow:Can save best model only with val_auc available, skipping.
    roc-auc_train: 0.5321 - roc-auc_val: 0.5312
    670/670 [==============================] - 2s 3ms/step - loss: 0.6881 - val_loss:

    Epoch 00012: LearningRateScheduler setting learning rate to 0.008145062532275914.
    Epoch 12/20
    660/670 [============================>.] - ETA: 0s - loss: 0.6879  f1_score: 0.527
    WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
    WARNING:tensorflow:Can save best model only with val_auc available, skipping.
    roc-auc_train: 0.5322 - roc-auc_val: 0.5321
    670/670 [==============================] - 1s 2ms/step - loss: 0.6879 - val_loss:

    Epoch 00013: LearningRateScheduler setting learning rate to 0.008145062252879143.
    Epoch 13/20
    649/670 [============================>.] - ETA: 0s - loss: 0.6875  f1_score: 0.525
```

```
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.5321 - roc-auc_val: 0.5323
670/670 [==============================] - 2s 3ms/step - loss: 0.6876 - val_loss: (

Epoch 00014: LearningRateScheduler setting learning rate to 0.008145062252879143.
Epoch 14/20
638/670 [===========================>..] - ETA: 0s - loss: 0.6871  f1_score: 0.527
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.5296 - roc-auc_val: 0.5314
670/670 [==============================] - 2s 3ms/step - loss: 0.6873 - val_loss:
```

Model 2 Observations:

1. Epoch No. 20 given Maximum F1 Score: 0.5358 & roc-auc_val: 0.5343
2. As Epoch number increases, loss decreases

```
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.
```

Saved successfully! ×

```
#Input layer
input_layer = Input(shape=(2,))
#Dense hidden layer
layer1 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform())(
#output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.he_uniform(
#Creating a model
model = Model(inputs=input_layer,outputs=output)

lrschedule = LearningRateScheduler(scheduler, verbose=0.1)
filepath="D:\Applied AI Course\Assignments\20. Assignment- Working with Callbacks\model_sa
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_o
earlystop = EarlyStopping(monitor='val_auc', patience=2, verbose=1, mode='max')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.0001, v


# here we are creating a list with all the callbacks we want
callback_list = [history_own,lrschedule, earlystop, checkpoint,terminate]
```

```
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.09, nesterov=False, nam

auc=tf.keras.metrics.AUC()

model.compile(optimizer=optimizer, loss='BinaryCrossentropy')

model.fit(X_train,y_train,epochs=20,validation_data=(X_test,y_test),batch_size=20,callback
```

```
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6515 - roc-auc_val: 0.6531
670/670 [==============================] - 3s 4ms/step - loss: 0.6683 - val_loss:

Epoch 00004: LearningRateScheduler setting learning rate to 0.009499999694526196.
Epoch 4/20
657/670 [============================>.] - ETA: 0s - loss: 0.6656  f1_score: 0.612
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6597 - roc-auc_val: 0.6612
670/670 [==============================] - 2s 4ms/step - loss: 0.6655 - val_loss:

Epoch 00005: LearningRateScheduler setting learning rate to 0.009499999694526196.
Epoch 5/20
669/670 [============================>.] - ETA: 0s - loss: 0.6626  f1_score: 0.623
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6684 - roc-auc_val: 0.6692
670/670 [==============================] - 2s 4ms/step - loss: 0.6626 - val_loss:

Epoch 00006: LearningRateScheduler setting learning rate to 0.009024999709799886.
Epoch 6/20
651/670 [============================>.] - ETA: 0s - loss: 0.6591  f1_score: 0.632
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
                                           t model only with val_auc available, skipping.
                                           _val: 0.6775
                                           ========] - 2s 4ms/step - loss: 0.6592 - val_loss:

Epoch 00007: LearningRateScheduler setting learning rate to 0.009025000035762787.
Epoch 7/20

641/670 [===========================>..] - ETA: 0s - loss: 0.6560  f1_score: 0.631
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6859 - roc-auc_val: 0.6847
670/670 [==============================] - 3s 4ms/step - loss: 0.6558 - val_loss:

Epoch 00008: LearningRateScheduler setting learning rate to 0.009025000035762787.
Epoch 8/20
664/670 [============================>.] - ETA: 0s - loss: 0.6521  f1_score: 0.633
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.6965 - roc-auc_val: 0.6939
670/670 [==============================] - 3s 4ms/step - loss: 0.6521 - val_loss:

Epoch 00009: LearningRateScheduler setting learning rate to 0.008573750033974648.
Epoch 9/20
649/670 [============================>.] - ETA: 0s - loss: 0.6487  f1_score: 0.638
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
roc-auc_train: 0.7047 - roc-auc_val: 0.701
670/670 [==============================] - 2s 3ms/step - loss: 0.6482 - val_loss:
```

Saved successfully!

```
Epoch 00010: LearningRateScheduler setting learning rate to 0.008573750033974648.
Epoch 10/20
654/670 [============================>.] - ETA: 0s - loss: 0.6448  f1_score: 0.645
WARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not ava
WARNING:tensorflow:Can save best model only with val_auc available, skipping.
```

Model 3 Observations:

1. Epoch No. 20 given Maximum F1 Score: 0.6586 & roc-auc_val: 0.723
2. Initially, as epoch number increases, F1 score & roc-auc_val increases,
3. As Epoch number increases, val_loss decreases

Model-4

```
1. Try with any values to get better accuracy/f1 score.
```

Saved successfully! ✕

```
#Input layer
input layer = Input(shape=(2,))
ernel_initializer=tf.keras.initializers.he_uniform())(
#output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.he_uniform(
#Creating a model
model = Model(inputs=input_layer,outputs=output)

lrschedule = LearningRateScheduler(scheduler, verbose=0.1)
filepath="D:\Applied AI Course\Assignments\20. Assignment- Working with Callbacks\model_sa
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_o
earlystop = EarlyStopping(monitor='val_auc', patience=2, verbose=1, mode='max')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.0001, v

# here we are creating a list with all the callbacks we want
callback_list = [history_own,lrschedule, earlystop, checkpoint,terminate]

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.09, nesterov=False, nam

auc=tf.keras.metrics.AUC()

model.compile(optimizer=optimizer, loss='BinaryCrossentropy')

model.fit(X_train,y_train,epochs=20,validation_data=(X_test,y_test),batch_size=20,callback
```

```
roc-auc_train: 0.6157 - roc-auc_val: 0.6203
670/670 [==============================] - 3s 4ms/step - loss: 0.6782 - val loss: 0
```

```
:poch 00007: LearningRateScheduler setting learning rate to 0.009025000035762787.
:poch 7/20
566/670 [============================>.] - ETA: 0s - loss: 0.6763  f1_score: 0.5933
NARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not avai
NARNING:tensorflow:Can save best model only with val_auc available, skipping.
·oc-auc_train: 0.6187 - roc-auc_val: 0.6237
570/670 [==============================] - 1s 2ms/step - loss: 0.6763 - val_loss: 0

:poch 00008: LearningRateScheduler setting learning rate to 0.009025000035762787.
:poch 8/20
530/670 [===========================>..] - ETA: 0s - loss: 0.6753  f1_score: 0.5909
NARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not avai
NARNING:tensorflow:Can save best model only with val_auc available, skipping.
·oc-auc_train: 0.6179 - roc-auc_val: 0.6226
570/670 [==============================] - 2s 3ms/step - loss: 0.6747 - val_loss: 0

:poch 00009: LearningRateScheduler setting learning rate to 0.008573750033974648.
:poch 9/20
540/670 [===========================>..] - ETA: 0s - loss: 0.6733  f1_score: 0.5859
NARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not avai
NARNING:tensorflow:Can save best model only with val_auc available, skipping.
·oc-auc_train: 0.6165 - roc-auc_val: 0.6215
570/670 [==============================] - 2s 3ms/step - loss: 0.6731 - val_loss: 0

:poch 00010: LearningRateScheduler setting learning rate to 0.008573750033974648.
:poch 10/20
553/670 [============================>.] - ETA: 0s - loss: 0.6719  f1_score: 0.5883
NARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not avai
NARNING:tensorflow:Can save best model only with val_auc available, skipping.
·oc-auc_train: 0.6195 - roc-auc_val: 0.6239
570/670 [==============================] - 1s 2ms/step - loss: 0.6718 - val_loss: 0

Saved successfully!                    er setting learning rate to 0.008573750033974648.

564/670 [============================>.] - ETA: 0s - loss: 0.6704  f1_score: 0.5900
NARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not avai
NARNING:tensorflow:Can save best model only with val_auc available, skipping.
·oc-auc_train: 0.6214 - roc-auc_val: 0.6263
570/670 [==============================] - 2s 2ms/step - loss: 0.6705 - val_loss: 0

:poch 00012: LearningRateScheduler setting learning rate to 0.008145062532275914.
:poch 12/20
540/670 [===========================>..] - ETA: 0s - loss: 0.6696  f1_score: 0.5911
NARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not avai
NARNING:tensorflow:Can save best model only with val_auc available, skipping.
·oc-auc_train: 0.6245 - roc-auc_val: 0.6292
570/670 [==============================] - 2s 2ms/step - loss: 0.6692 - val_loss: 0

:poch 00013: LearningRateScheduler setting learning rate to 0.008145062252879143.
:poch 13/20
532/670 [===========================>..] - ETA: 0s - loss: 0.6671  f1_score: 0.5892
NARNING:tensorflow:Early stopping conditioned on metric `val_auc` which is not avai
NARNING:tensorflow:Can save best model only with val_auc available, skipping.
·oc-auc_train: 0.6281 - roc-auc_val: 0.6324
570/670 [==============================] - 1s 2ms/step - loss: 0.6677 - val_loss: 0
```

Model 4 Observations:

1. Epoch No. 20 given Maximum F1 Score: 0.6155 & roc-auc_val: 0.6712
2. As Epoch number increases, val_loss decreases

✓ 41s completed at 6:59 PM ● ×

Saved successfully! ×