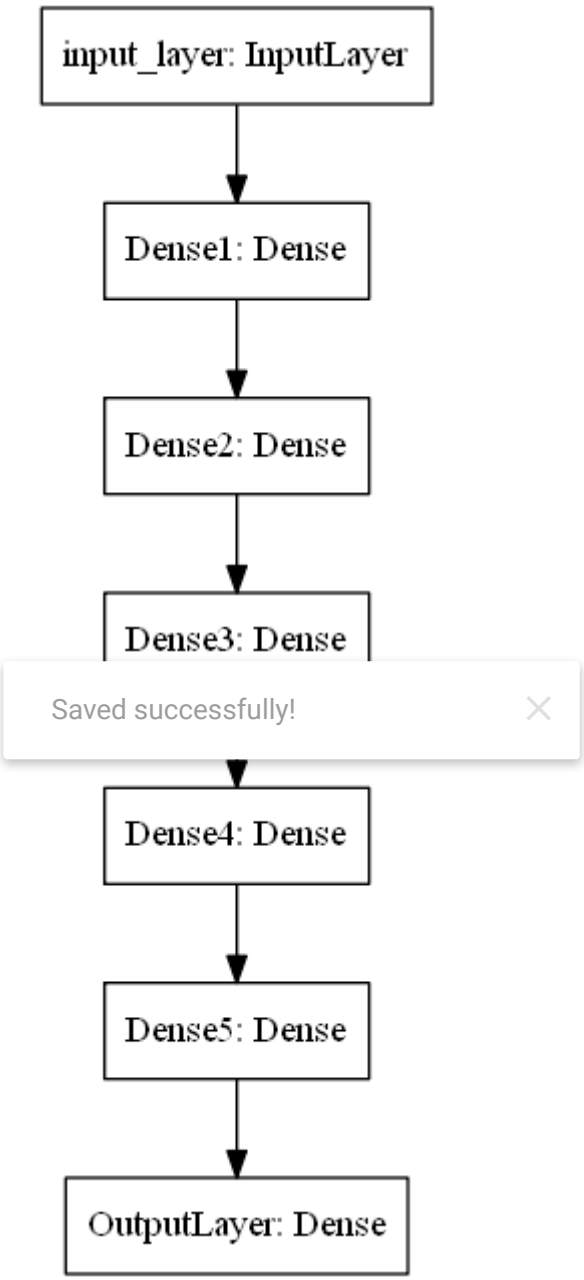
1. Download the data from here. You have to use data.csv file for this assignment

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




Saved successfully!

## 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 He initialization as initializer.
4. 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.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
```

```
data=pd.read_csv("data.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!

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

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)

    #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=0.0, nesterov=False, name

    model.compile(optimizer=optimizer,loss='BinaryCrossentropy',metrics=['AUC'])
```

Saved successfully!

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

```
    Epoch 1/5
    663/670 [============================>.] - ETA: 0s - loss: 0.7015 - auc: 0.5003  f1_s
    670/670 [==============================] - 2s 3ms/step - loss: 0.7014 - auc: 0.5003 -
    Epoch 2/5
    657/670 [============================>.] - ETA: 0s - loss: 0.6934 - auc: 0.5126  f1_s
    670/670 [==============================] - 2s 2ms/step - loss: 0.6934 - auc: 0.5134 -
    Epoch 3/5
    654/670 [============================>.] - ETA: 0s - loss: 0.6928 - auc: 0.5142  f1_s
    670/670 [==============================] - 2s 3ms/step - loss: 0.6928 - auc: 0.5143 -
    Epoch 4/5
    654/670 [============================>.] - ETA: 0s - loss: 0.6925 - auc: 0.5247  f1_s
    670/670 [==============================] - 3s 4ms/step - loss: 0.6925 - auc: 0.5254 -
    Epoch 5/5
    669/670 [============================>.] - ETA: 0s - loss: 0.6923 - auc: 0.5311  f1_s
    670/670 [==============================] - 2s 4ms/step - loss: 0.6923 - auc: 0.5309 -
    <keras.callbacks.History at 0x7fad0fe31390>
```

```
    history_own.val_f1s
```

```
    [0.5077272727272727,
     0.4860606060606061,
     0.5013636363636363,
     0.5075757575757576,
     0.5209090909090909]
```

```
#If you are getting any NaN values(either weigths or loss) while training, you have to ter
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



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 [==============================] - 2s 3ms/step - loss: 0.6921 - auc: 0.5437 -
Epoch 2/5
670/670 [==============================] - 2s 3ms/step - loss: 0.6918 - auc: 0.5506 -
Epoch 3/5
670/670 [==============================] - 2s 3ms/step - loss: 0.6915 - auc: 0.5587 -
Epoch 4/5
670/670 [==============================] - 2s 3ms/step - loss: 0.6912 - auc: 0.5590 -
Epoch 5/5
670/670 [==============================] - 3s 4ms/step - loss: 0.6909 - auc: 0.5636 -
<keras.callbacks.History at 0x7fad0de8ca50>
```



```
our 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
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_o

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, name

model.compile(optimizer=optimizer, loss='BinaryCrossentropy',metrics=['AUC'])

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

```
Epoch 1/5
668/670 [============================>.] - ETA: 0s - loss: 0.7350 - auc: 0.4993
```

```
Epoch 00001: val_auc improved from -inf to 0.49661, saving model to D:\Applied AI Cou
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 3s 3ms/step - loss: 0.7349 - auc: 0.4995 -
Epoch 2/5
652/670 [============================>.] - ETA: 0s - loss: 0.6935 - auc: 0.5020
Epoch 00002: val_auc improved from 0.49661 to 0.53466, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 3ms/step - loss: 0.6935 - auc: 0.5024 -
Epoch 3/5
647/670 [===========================>..] - ETA: 0s - loss: 0.6921 - auc: 0.5388
Epoch 00003: val_auc improved from 0.53466 to 0.55845, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 3ms/step - loss: 0.6921 - auc: 0.5404 -
Epoch 4/5
651/670 [============================>.] - ETA: 0s - loss: 0.6916 - auc: 0.5630
Epoch 00004: val_auc improved from 0.55845 to 0.57388, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 3ms/step - loss: 0.6916 - auc: 0.5631 -
Epoch 5/5
658/670 [============================>.] - ETA: 0s - loss: 0.6910 - auc: 0.5807
Epoch 00005: val_auc improved from 0.57388 to 0.59294, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 3ms/step - loss: 0.6910 - auc: 0.5805 -
<keras.callbacks.History at 0x7fad0de8e450>
```

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



```
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=0.0, nesterov=False, name

model.compile(optimizer=optimizer, loss='BinaryCrossentropy',metrics=['AUC'])


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

```
Epoch 1/5
670/670 [==============================] - 2s 2ms/step - loss: 0.7426 - auc: 0.4988 -
Epoch 2/5
670/670 [==============================] - 2s 2ms/step - loss: 0.6992 - auc: 0.5014 -
Epoch 3/5
670/670 [==============================] - 1s 2ms/step - loss: 0.6941 - auc: 0.5083 -
Epoch 00003: early stopping
<keras.callbacks.History at 0x7fad0d5e3f90>
```

```
#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%.

def changeLearningRate(epoch):
    initial_learningrate=0.01
    if epoch % 3 ==0:
      changed = initial_learningrate*(1-0.05)**epoch
    else:
      changed = initial_learningrate*(1-0.1)**epoch
    return changed

changed_lr = []
for i in range(1,10):
  changed_lr.append(changeLearningRate(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(changeLearningRate, verbose=0.1)
filepath="model_save/weights-{epoch:02d}-{val_auc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_o
earlystop = EarlyStopping(monitor='val_auc', patience=2, verbose=1, mode='max')

# 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=0.0, nesterov=False, name

model.compile(optimizer=optimizer, loss='BinaryCrossentropy',metrics=['AUC'])


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 0.01.
     Epoch 1/20
     668/670 [============================>.] - ETA: 0s - loss: 0.7179 - auc: 0.5021  f1_s

     Epoch 00001: val_auc improved from -inf to 0.50129, saving model to model_save/weight
     670/670 [==============================] - 2s 3ms/step - loss: 0.7179 - auc: 0.5019 -

     Epoch 00002: LearningRateScheduler setting learning rate to 0.009000000000000001.
     Epoch 2/20
     664/670 [============================>.] - ETA: 0s - loss: 0.6930 - auc: 0.5212  f1_s

     Epoch 00002: val_auc improved from 0.50129 to 0.52587, saving model to model_save/wei
     670/670 [==============================] - 2s 3ms/step - loss: 0.6929 - auc: 0.5218 -
```

Saved successfully!

```
Epoch 00003: LearningRateScheduler setting learning rate to 0.008100000000000001.
Epoch 3/20
653/670 [============================>.] - ETA: 0s - loss: 0.6915 - auc: 0.5507  f1_s

Epoch 00003: val_auc improved from 0.52587 to 0.56857, saving model to model_save/wei
670/670 [==============================] - 2s 3ms/step - loss: 0.6914 - auc: 0.5513 -

Epoch 00004: LearningRateScheduler setting learning rate to 0.00857375.
Epoch 4/20
637/670 [===========================>..] - ETA: 0s - loss: 0.6905 - auc: 0.5777  f1_s

Epoch 00004: val_auc improved from 0.56857 to 0.60840, saving model to model_save/wei
670/670 [==============================] - 2s 3ms/step - loss: 0.6905 - auc: 0.5775 -

Epoch 00005: LearningRateScheduler setting learning rate to 0.006561.
Epoch 5/20
648/670 [============================>.] - ETA: 0s - loss: 0.6898 - auc: 0.6032  f1_s

Epoch 00005: val_auc improved from 0.60840 to 0.62230, saving model to model_save/wei
670/670 [==============================] - 2s 2ms/step - loss: 0.6898 - auc: 0.6025 -

Epoch 00006: LearningRateScheduler setting learning rate to 0.005904900000000001.
Epoch 6/20
639/670 [===========================>..] - ETA: 0s - loss: 0.6893 - auc: 0.6111  f1_s

Epoch 00006: val_auc did not improve from 0.62230
670/670 [==============================] - 2s 3ms/step - loss: 0.6893 - auc: 0.6130 -

Epoch 00007: LearningRateScheduler setting learning rate to 0.007350918906249998.
                                                    ======>.] - ETA: 0s - loss: 0.6886 - auc: 0.6073  f1_s

Epoch 00007: val_auc did not improve from 0.62230
670/670 [==============================] - 2s 3ms/step - loss: 0.6886 - auc: 0.6073 -
Epoch 00007: early stopping
<keras.callbacks.History at 0x7fad0eb40190>
```

Saved successfully!

***Model 1 Observations:***

1. Epoch No. 6 given Maximum F1 Score: 0.6138 & val_auc= 0.6222
2. Initially, val_auc increases gradually, after Epoch number 5 val_auc is not increasing
3. As Epoch number increases, val_loss decreases

**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.

```
#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(changeLearningRate, 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')

# 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=0.0, nesterov=False, name

model.compile(optimizer=optimizer, loss='BinaryCrossentropy',metrics=['AUC'])


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

Saved successfully!

```
ler setting learning rate to 0.01.
======>.] - ETA: 0s - loss: 0.7053 - auc: 0.4446  f1_s

Epoch 00001: val_auc improved from -inf to 0.39784, saving model to D:\Applied AI Cou
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 3s 4ms/step - loss: 0.7053 - auc: 0.4441 -

Epoch 00002: LearningRateScheduler setting learning rate to 0.009000000000000001.
Epoch 2/20
659/670 [============================>.] - ETA: 0s - loss: 0.6941 - auc: 0.4696  f1_s

Epoch 00002: val_auc improved from 0.39784 to 0.51278, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 4ms/step - loss: 0.6940 - auc: 0.4712 -

Epoch 00003: LearningRateScheduler setting learning rate to 0.008100000000000001.
Epoch 3/20
639/670 [===========================>..] - ETA: 0s - loss: 0.6921 - auc: 0.5174  f1_s

Epoch 00003: val_auc improved from 0.51278 to 0.53137, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 3ms/step - loss: 0.6921 - auc: 0.5193 -

Epoch 00004: LearningRateScheduler setting learning rate to 0.00857375.
Epoch 4/20
649/670 [============================>.] - ETA: 0s - loss: 0.6913 - auc: 0.5228  f1_s

Epoch 00004: val_auc did not improve from 0.53137
670/670 [==============================] - 2s 3ms/step - loss: 0.6914 - auc: 0.5213 -
```

```
Epoch 00005: LearningRateScheduler setting learning rate to 0.006561.
Epoch 5/20
667/670 [============================>.] - ETA: 0s - loss: 0.6911 - auc: 0.5223  f1_s

Epoch 00005: val_auc did not improve from 0.53137
670/670 [==============================] - 2s 3ms/step - loss: 0.6910 - auc: 0.5232 -
Epoch 00005: early stopping
<keras.callbacks.History at 0x7fad0dee6b10>
```

***Model 2 Observations:***

1. Epoch No. 5 given Maximum F1 Score: 0.5215 & val_auc= 0.5290
2. Initially, val_auc increases gradually, after Epoch number 3 val_auc is not increasing
3. As Epoch number increases, val_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

Saved successfully!

ng process.

```
#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(changeLearningRate, 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')

# 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=0.0, nesterov=False, name

model.compile(optimizer=optimizer, loss='BinaryCrossentropy',metrics=['AUC'])
```

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

```
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- l
670/670 [==============================] - 2s 3ms/step - loss: 0.6643 - auc: 0.646

Epoch 00014: LearningRateScheduler setting learning rate to 0.002541865828329001.
Epoch 14/20
660/670 [============================>.] - ETA: 0s - loss: 0.6633 - auc: 0.6493  f

Epoch 00014: val_auc improved from 0.64768 to 0.64970, saving model to D:\Applied A
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- l
670/670 [==============================] - 2s 3ms/step - loss: 0.6632 - auc: 0.649

Epoch 00015: LearningRateScheduler setting learning rate to 0.002287679245496101.
Epoch 15/20
655/670 [============================>.] - ETA: 0s - loss: 0.6621 - auc: 0.6526  f

Epoch 00015: val_auc improved from 0.64970 to 0.65191, saving model to D:\Applied A
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- l
670/670 [==============================] - 2s 4ms/step - loss: 0.6625 - auc: 0.651

Epoch 00016: LearningRateScheduler setting learning rate to 0.00463291230159753.
Epoch 16/20
661/670 [============================>.] - ETA: 0s - loss: 0.6616 - auc: 0.6542  f

Epoch 00016: val_auc improved from 0.65191 to 0.65601, saving model to D:\Applied A
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- l
========] - 2s 3ms/step - loss: 0.6615 - auc: 0.654

ler setting learning rate to 0.0018530201888518416.
Epoch 17/20
660/670 [============================>.] - ETA: 0s - loss: 0.6603 - auc: 0.6580  f

Epoch 00017: val_auc improved from 0.65601 to 0.65775, saving model to D:\Applied A
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- l
670/670 [==============================] - 2s 4ms/step - loss: 0.6604 - auc: 0.657

Epoch 00018: LearningRateScheduler setting learning rate to 0.0016677181699666576.
Epoch 18/20
651/670 [============================>.] - ETA: 0s - loss: 0.6599 - auc: 0.6596  f

Epoch 00018: val_auc improved from 0.65775 to 0.65930, saving model to D:\Applied A
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- l
670/670 [==============================] - 2s 3ms/step - loss: 0.6599 - auc: 0.659

Epoch 00019: LearningRateScheduler setting learning rate to 0.003972143184582182.
Epoch 19/20
669/670 [============================>.] - ETA: 0s - loss: 0.6591 - auc: 0.6616  f

Epoch 00019: val_auc improved from 0.65930 to 0.66312, saving model to D:\Applied A
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- l
670/670 [==============================] - 2s 3ms/step - loss: 0.6590 - auc: 0.661

Epoch 00020: LearningRateScheduler setting learning rate to 0.0013508517176729928.
Epoch 20/20
658/670 [============================>.] - ETA: 0s - loss: 0.6580 - auc: 0.6652  f

Epoch 00020: val_auc improved from 0.66312 to 0.66447, saving model to D:\Applied A
```

Saved successfully!

```
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- L
```

***Model 3 Observations:***

1. Epoch No. 20 given Maximum F1 Score: 0.6244 & val_auc= 0.6645
2. Initially, as epoch number increases, val_auc is also increases
3. As Epoch number increases, val_loss decreases

**Model-4**

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

```
#Input layer
input_layer = Input(shape=(2,))
#Dense hidden layer
ernel_initializer=tf.keras.initializers.he_uniform())(

output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.he_uniform(
#Creating a model
model = Model(inputs=input_layer,outputs=output)

lrschedule = LearningRateScheduler(changeLearningRate, 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')

# 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=0.0, nesterov=False, name

model.compile(optimizer=optimizer, loss='BinaryCrossentropy',metrics=['AUC'])


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

Saved successfully!

```
Epoch 00001: LearningRateScheduler setting learning rate to 0.01.
Epoch 1/20
636/670 [===========================>..] - ETA: 0s - loss: 0.6953 - auc: 0.5119  f1_s

Epoch 00001: val_auc improved from -inf to 0.57145, saving model to D:\Applied AI Cou
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
```

```
670/670 [==============================] - 3s 4ms/step - loss: 0.6951 - auc: 0.5130 -

Epoch 00002: LearningRateScheduler setting learning rate to 0.009000000000000001.
Epoch 2/20
649/670 [============================>.] - ETA: 0s - loss: 0.6889 - auc: 0.5763  f1_s

Epoch 00002: val_auc improved from 0.57145 to 0.61152, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 4ms/step - loss: 0.6891 - auc: 0.5749 -

Epoch 00003: LearningRateScheduler setting learning rate to 0.008100000000000001.
Epoch 3/20
658/670 [============================>.] - ETA: 0s - loss: 0.6866 - auc: 0.5949  f1_s

Epoch 00003: val_auc did not improve from 0.61152
670/670 [==============================] - 2s 2ms/step - loss: 0.6865 - auc: 0.5961 -

Epoch 00004: LearningRateScheduler setting learning rate to 0.00857375.
Epoch 4/20
653/670 [============================>.] - ETA: 0s - loss: 0.6846 - auc: 0.6118  f1_s

Epoch 00004: val_auc improved from 0.61152 to 0.61500, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 3ms/step - loss: 0.6846 - auc: 0.6112 -

Epoch 00005: LearningRateScheduler setting learning rate to 0.006561.
Epoch 5/20
669/670 [============================>.] - ETA: 0s - loss: 0.6827 - auc: 0.6170  f1_s

prove from 0.61500
========] - 1s 2ms/step - loss: 0.6827 - auc: 0.6169 -

Epoch 00006: LearningRateScheduler setting learning rate to 0.005904900000000001.
Epoch 6/20
666/670 [============================>.] - ETA: 0s - loss: 0.6815 - auc: 0.6173  f1_s

Epoch 00006: val_auc did not improve from 0.61500
670/670 [==============================] - 1s 2ms/step - loss: 0.6815 - auc: 0.6176 -
Epoch 00006: early stopping
<keras.callbacks.History at 0x7fad0e942850>
```

Saved successfully!

***Model 4 Observations:***

1. Epoch No. 4 given Maximum F1 Score: 0.5892 & val_auc= 0.6150
2. As Epoch number increases, val_loss decreases

✓ 12s completed at 2:07 AM

Saved successfully!