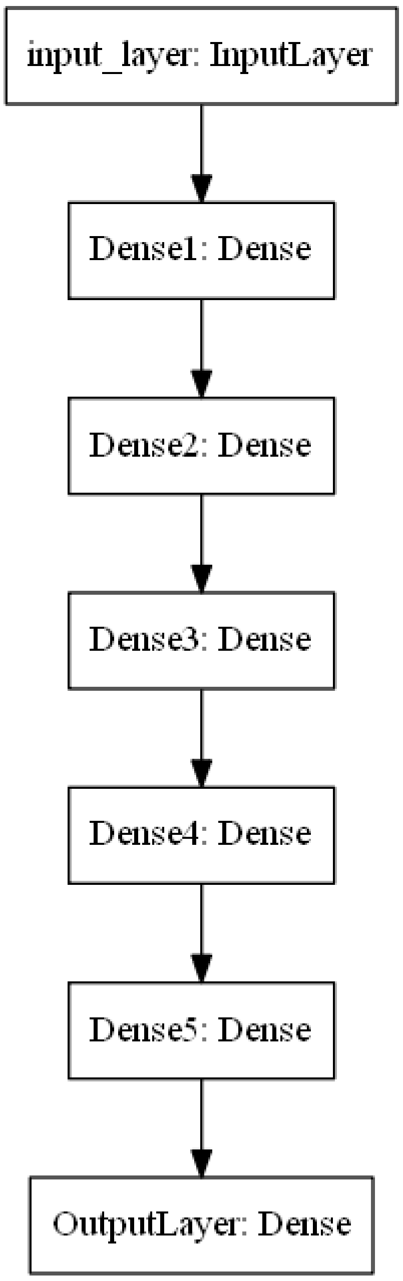
1. Download the data from here. You have to use data.csv file for this assignment

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.
```

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
from keras.callbacks import ReduceLROnPlateau
```

```
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

```
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=0.09, 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', metrics=[tf.keras.metrics.AU

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

```
Epoch 1/5
654/670 [============================>.] - ETA: 0s - loss: 0.7442 - auc: 0.4946  f1_s
670/670 [==============================] - 3s 3ms/step - loss: 0.7431 - auc: 0.4947 -
Epoch 2/5
669/670 [============================>.] - ETA: 0s - loss: 0.6946 - auc: 0.4970  f1_s
670/670 [==============================] - 2s 3ms/step - loss: 0.6946 - auc: 0.4970 -
Epoch 3/5
654/670 [============================>.] - ETA: 0s - loss: 0.6932 - auc: 0.5071  f1_s
670/670 [==============================] - 2s 2ms/step - loss: 0.6933 - auc: 0.5064 -
Epoch 4/5
660/670 [============================>.] - ETA: 0s - loss: 0.6930 - auc: 0.5158  f1_s
670/670 [==============================] - 2s 3ms/step - loss: 0.6930 - auc: 0.5148 -
Epoch 5/5
640/670 [===========================>..] - ETA: 0s - loss: 0.6927 - auc: 0.5238  f1_s
670/670 [==============================] - 2s 3ms/step - loss: 0.6927 - auc: 0.5241 -
<keras.callbacks.History at 0x7f9e467fee90>
```

```
history_own.val_f1s
```

```
[0.5124242424242424,
 0.5177272727272727,
 0.5134848484848484,
 0.5272727272727272,
 0.5357575757575758]
```

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 [==============================] - 2s 2ms/step - loss: 0.6925 - auc: 0.5336 ·
Epoch 2/5
670/670 [==============================] - 1s 2ms/step - loss: 0.6921 - auc: 0.5455 ·
Epoch 3/5
670/670 [==============================] - 1s 2ms/step - loss: 0.6917 - auc: 0.5623 ·
Epoch 4/5
670/670 [==============================] - 2s 2ms/step - loss: 0.6913 - auc: 0.5623 ·
Epoch 5/5
670/670 [==============================] - 1s 2ms/step - loss: 0.6908 - auc: 0.5708 ·
<keras.callbacks.History at 0x7f9e46597510>
```

```
#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
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', metrics=[tf.keras.metrics.AU

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

```
Epoch 1/5
669/670 [============================>.] - ETA: 0s - loss: 0.7117 - auc: 0.4943
Epoch 00001: val_auc improved from -inf to 0.50576, 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: 0.7117 - auc: 0.4942 -
Epoch 2/5
642/670 [===========================>..] - ETA: 0s - loss: 0.6935 - auc: 0.5026
Epoch 00002: val_auc did not improve from 0.50576
670/670 [==============================] - 2s 2ms/step - loss: 0.6935 - auc: 0.5010 -
Epoch 3/5
660/670 [============================>.] - ETA: 0s - loss: 0.6931 - auc: 0.5046
Epoch 00003: val_auc improved from 0.50576 to 0.50914, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 2s 3ms/step - loss: 0.6931 - auc: 0.5046 -
Epoch 4/5
647/670 [===========================>..] - ETA: 0s - loss: 0.6930 - auc: 0.5067
Epoch 00004: val_auc did not improve from 0.50914

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0019999999552965165.
670/670 [==============================] - 2s 2ms/step - loss: 0.6930 - auc: 0.5075 -
Epoch 5/5
663/670 [============================>.] - ETA: 0s - loss: 0.6928 - auc: 0.5180
Epoch 00005: val_auc did not improve from 0.50914
670/670 [==============================] - 2s 2ms/step - loss: 0.6928 - auc: 0.5176 -
<keras.callbacks.History at 0x7f9e464c38d0>
```

```
#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=0.0, nesterov=False, name

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', metrics=[tf.keras.metrics.AU

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: 0.7539 - auc: 0.4922 ·
Epoch 2/5
670/670 [==============================] - 1s 2ms/step - loss: 0.6979 - auc: 0.4965 ·
Epoch 3/5
670/670 [==============================] - 2s 2ms/step - loss: 0.6948 - auc: 0.4933 ·
Epoch 4/5
670/670 [==============================] - 1s 2ms/step - loss: 0.6939 - auc: 0.4933 ·
Epoch 5/5
670/670 [==============================] - 2s 2ms/step - loss: 0.6935 - auc: 0.5001 ·
<keras.callbacks.History at 0x7f9e35181750>
```

```
#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+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))

#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')
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=0.0, nesterov=False, name


auc=tf.keras.metrics.AUC()

model.compile(optimizer=optimizer, loss='BinaryCrossentropy', metrics=[tf.keras.metrics.AU

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

```
Epoch 00012: val_auc improved from 0.63198 to 0.64358, saving model to model_save/
670/670 [==============================] - 2s 3ms/step - loss: 0.6888 - auc: 0.625

Epoch 00013: LearningRateScheduler setting learning rate to 0.002541865828329001.
Epoch 13/20
647/670 [===========================>..] - ETA: 0s - loss: 0.6884 - auc: 0.6432  f

Epoch 00013: val_auc improved from 0.64358 to 0.65050, saving model to model_save/
670/670 [==============================] - 2s 2ms/step - loss: 0.6885 - auc: 0.641

Epoch 00014: LearningRateScheduler setting learning rate to 0.002287679245496101.
Epoch 14/20
653/670 [============================>.] - ETA: 0s - loss: 0.6883 - auc: 0.6505  f

Epoch 00014: val_auc did not improve from 0.65050
670/670 [==============================] - 2s 3ms/step - loss: 0.6882 - auc: 0.650

Epoch 00015: LearningRateScheduler setting learning rate to 0.00463291230159753.
Epoch 15/20
649/670 [============================>.] - ETA: 0s - loss: 0.6879 - auc: 0.6453  f

Epoch 00015: val_auc improved from 0.65050 to 0.65806, saving model to model_save/
670/670 [==============================] - 2s 3ms/step - loss: 0.6879 - auc: 0.645

Epoch 00016: LearningRateScheduler setting learning rate to 0.0018530201888518416.
Epoch 16/20
664/670 [============================>.] - ETA: 0s - loss: 0.6876 - auc: 0.6529  f

Epoch 00016: val_auc improved from 0.65806 to 0.65899, saving model to model_save/
670/670 [==============================] - 2s 3ms/step - loss: 0.6876 - auc: 0.652

Epoch 00017: LearningRateScheduler setting learning rate to 0.0016677181699666576.
Epoch 17/20
654/670 [============================>.] - ETA: 0s - loss: 0.6874 - auc: 0.6558  f

Epoch 00017: val_auc improved from 0.65899 to 0.66043, saving model to model_save/
670/670 [==============================] - 2s 3ms/step - loss: 0.6874 - auc: 0.655

Epoch 00018: LearningRateScheduler setting learning rate to 0.003972143184582182.
Epoch 18/20
642/670 [===========================>..] - ETA: 0s - loss: 0.6872 - auc: 0.6544  f

Epoch 00018: val_auc improved from 0.66043 to 0.66742, saving model to model_save/
670/670 [==============================] - 2s 3ms/step - loss: 0.6872 - auc: 0.654
```

```
Epoch 00019: LearningRateScheduler setting learning rate to 0.0013508517176729928.
Epoch 19/20
660/670 [============================>.] - ETA: 0s - loss: 0.6869 - auc: 0.6617  f

Epoch 00019: val_auc improved from 0.66742 to 0.66879, saving model to model_save/w
670/670 [==============================] - 2s 2ms/step - loss: 0.6869 - auc: 0.662

Epoch 00020: LearningRateScheduler setting learning rate to 0.0012157665459056935.
Epoch 20/20
652/670 [============================>.] - ETA: 0s - loss: 0.6867 - auc: 0.6658  f

Epoch 00020: val_auc improved from 0.66879 to 0.66986, saving model to model_save/w
670/670 [==============================]   2s 2ms/step   loss: 0.6867   auc: 0.665
```

Model 1 Observations:

1. Epoch No. 15 given Maximum F1 Score: 0.6273 & val_auc= 0.6875
2. As Epoch number increases, val_auc increases
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')
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', metrics=[tf.keras.metrics.AU

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 0.009000000000000001.
Epoch 1/20
661/670 [============================>.] - ETA: 0s - loss: 0.7257 - auc: 0.4542  f1_s

Epoch 00001: val_auc improved from -inf to 0.45477, 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: 0.7255 - auc: 0.4538 -

Epoch 00002: LearningRateScheduler setting learning rate to 0.008100000000000001.
Epoch 2/20
644/670 [===========================>..] - ETA: 0s - loss: 0.7006 - auc: 0.4182  f1_s

Epoch 00002: val_auc did not improve from 0.45477
670/670 [==============================] - 2s 3ms/step - loss: 0.7005 - auc: 0.4176 -

Epoch 00003: LearningRateScheduler setting learning rate to 0.00857375.
Epoch 3/20
649/670 [============================>.] - ETA: 0s - loss: 0.6941 - auc: 0.4692  f1_s

Epoch 00003: val_auc improved from 0.45477 to 0.51464, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 4s 5ms/step - loss: 0.6940 - auc: 0.4702 -

Epoch 00004: LearningRateScheduler setting learning rate to 0.006561.
Epoch 4/20
670/670 [==============================] - ETA: 0s - loss: 0.6917 - auc: 0.5296  f1_s

Epoch 00004: val_auc improved from 0.51464 to 0.52379, saving model to D:\Applied AI
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- Wor
670/670 [==============================] - 3s 5ms/step - loss: 0.6917 - auc: 0.5296 -

Epoch 00005: LearningRateScheduler setting learning rate to 0.005904900000000001.
Epoch 5/20
635/670 [===========================>..] - ETA: 0s - loss: 0.6908 - auc: 0.5347  f1_s

Epoch 00005: val_auc did not improve from 0.52379
670/670 [==============================] - 2s 3ms/step - loss: 0.6908 - auc: 0.5337 -

Epoch 00006: LearningRateScheduler setting learning rate to 0.007350918906249998.
Epoch 6/20
661/670 [============================>.] - ETA: 0s - loss: 0.6902 - auc: 0.5359  f1_s

Epoch 00006: val_auc did not improve from 0.52379
670/670 [==============================] - 2s 2ms/step - loss: 0.6902 - auc: 0.5358 -
Epoch 00006: early stopping
<keras.callbacks.History at 0x7f9e46341850>
```

Model 2 Observations:

1. Epoch No. 6 given Maximum F1 Score: 0.5108 & val_auc= 0.5230
2. 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.
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.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')
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', metrics=[tf.keras.metrics.AU

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
670/670 [==============================] - 2s 4ms/step - loss: 0.6582 - auc: 0.674

Epoch 00014: LearningRateScheduler setting learning rate to 0.002287679245496101.
Epoch 14/20
649/670 [============================>.] - ETA: 0s - loss: 0.6569 - auc: 0.6793  f

Epoch 00014: val_auc improved from 0.67983 to 0.68255, saving model to D:\Applied
```

```
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- \
670/670 [==============================] - 3s 4ms/step - loss: 0.6571 - auc: 0.678

Epoch 00015: LearningRateScheduler setting learning rate to 0.00463291230159753.
Epoch 15/20
669/670 [============================>.] - ETA: 0s - loss: 0.6555 - auc: 0.6826  f

Epoch 00015: val_auc improved from 0.68255 to 0.68834, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- \
670/670 [==============================] - 3s 4ms/step - loss: 0.6556 - auc: 0.682

Epoch 00016: LearningRateScheduler setting learning rate to 0.0018530201888518416.
Epoch 16/20
662/670 [============================>.] - ETA: 0s - loss: 0.6542 - auc: 0.6863  f

Epoch 00016: val_auc improved from 0.68834 to 0.69028, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- \
670/670 [==============================] - 4s 6ms/step - loss: 0.6540 - auc: 0.686

Epoch 00017: LearningRateScheduler setting learning rate to 0.0016677181699666576.
Epoch 17/20
670/670 [==============================] - ETA: 0s - loss: 0.6532 - auc: 0.6887  f

Epoch 00017: val_auc improved from 0.69028 to 0.69235, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- \
670/670 [==============================] - 3s 4ms/step - loss: 0.6532 - auc: 0.688

Epoch 00018: LearningRateScheduler setting learning rate to 0.003972143184582182.
Epoch 18/20
656/670 [============================>.] - ETA: 0s - loss: 0.6521 - auc: 0.6920  f

Epoch 00018: val_auc improved from 0.69235 to 0.69718, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- \
670/670 [==============================] - 3s 4ms/step - loss: 0.6520 - auc: 0.692

Epoch 00019: LearningRateScheduler setting learning rate to 0.0013508517176729928.
Epoch 19/20
641/670 [===========================>..] - ETA: 0s - loss: 0.6507 - auc: 0.6952  f

Epoch 00019: val_auc improved from 0.69718 to 0.69870, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- \
670/670 [==============================] - 2s 4ms/step - loss: 0.6507 - auc: 0.695

Epoch 00020: LearningRateScheduler setting learning rate to 0.0012157665459056935.
Epoch 20/20
665/670 [============================>.] - ETA: 0s - loss: 0.6502 - auc: 0.6965  f

Epoch 00020: val_auc improved from 0.69870 to 0.70005, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- \
670/670 [==============================] - 2s 3ms/step - loss: 0.6501 - auc: 0.697
```

Model 3 Observations:

1. Epoch No. 20 given Maximum F1 Score: 0.6455 & val_auc= 0.7000
2. Initially, as epoch number increases, F1 score & val_auc 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
layer1 = Dense(5,activation='selu',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')
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', metrics=[tf.keras.metrics.AU

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 0.009000000000000001.
Epoch 1/20
667/670 [============================>.] - ETA: 0s - loss: 0.7442 - auc: 0.5033  f

Epoch 00001: val_auc improved from -inf to 0.49381, saving model to D:\Applied AI 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- 
670/670 [==============================] - 3s 4ms/step - loss: 0.7440 - auc: 0.503

Epoch 00002: LearningRateScheduler setting learning rate to 0.008100000000000001.
Epoch 2/20
662/670 [============================>.] - ETA: 0s - loss: 0.6926 - auc: 0.5223  f

Epoch 00002: val_auc improved from 0.49381 to 0.53844, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- 
```

```
670/670 [==============================] - 2s 4ms/step - loss: 0.6926 - auc: 0.521

Epoch 00003: LearningRateScheduler setting learning rate to 0.00857375.
Epoch 3/20
658/670 [============================>.] - ETA: 0s - loss: 0.6903 - auc: 0.5549  f

Epoch 00003: val_auc improved from 0.53844 to 0.55578, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- 
670/670 [==============================] - 3s 4ms/step - loss: 0.6902 - auc: 0.555

Epoch 00004: LearningRateScheduler setting learning rate to 0.006561.
Epoch 4/20
669/670 [============================>.] - ETA: 0s - loss: 0.6875 - auc: 0.5798  f

Epoch 00004: val_auc improved from 0.55578 to 0.59607, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- 
670/670 [==============================] - 2s 3ms/step - loss: 0.6874 - auc: 0.580

Epoch 00005: LearningRateScheduler setting learning rate to 0.005904900000000001.
Epoch 5/20
667/670 [============================>.] - ETA: 0s - loss: 0.6854 - auc: 0.5986  f

Epoch 00005: val_auc improved from 0.59607 to 0.60118, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- 
670/670 [==============================] - 3s 4ms/step - loss: 0.6854 - auc: 0.598

Epoch 00006: LearningRateScheduler setting learning rate to 0.007350918906249998.
Epoch 6/20
656/670 [============================>.] - ETA: 0s - loss: 0.6839 - auc: 0.6061  f

Epoch 00006: val_auc improved from 0.60118 to 0.61226, saving model to D:\Applied 
INFO:tensorflow:Assets written to: D:\Applied AI Course\Assignments . Assignment- 
670/670 [==============================] - 3s 4ms/step - loss: 0.6840 - auc: 0.605

Epoch 00007: LearningRateScheduler setting learning rate to 0.004782969000000001.
Epoch 7/20
654/670 [============================>.] - ETA: 0s - loss: 0.6823 - auc: 0.6194  f

Epoch 00007: val_auc did not improve from 0.61226
670/670 [==============================] - 2s 3ms/step - loss: 0.6825 - auc: 0.617

Epoch 00008: LearningRateScheduler setting learning rate to 0.004304672100000001.
```

Model 4 Observations:

1. Epoch No. 13 given Maximum F1 Score: 0.5968 & val_auc= 0.6379
2. As Epoch number increases, val_loss decreases

✓ 41s completed at 3:02 AM