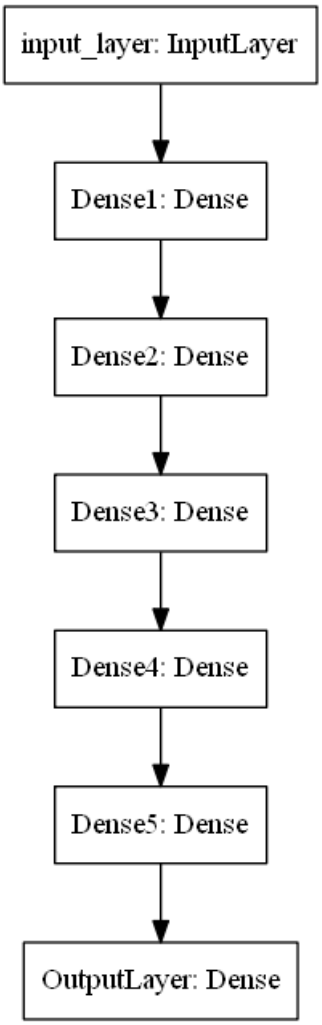
1. Download the data from here

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 ea ch 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, y ou 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 termina te 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 screensho ts for each model for evaluation)

9. use cross entropy as loss function

10. Try the architecture params as given below.

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.

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.

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.

Model-4

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

In [1]:

```
import numpy as np
import pandas as pd
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
import random as rn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from tensorflow.keras.callbacks import ModelCheckpoint
import tensorflow as tf
import datetime
import os
%load_ext tensorboard
```

loading data.csv file by downloading curlWget

1. Download the data from here

In [2]:

```
!wget --header="Host: doc-0o-58-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.104 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: https://drive.google.com/" --header="Cookie: AUTH_adlgb4krihkkkbgqd9sup8mmvt5it4ch_nonce=bdplkjva1bpi4" --header="Connection: keep-alive" "https://doc-0o-58-docs.googleusercontent.com/docs/securesc/aa6ktludmnluspcle3gr7p17lfha05rn/utonp9io8o9ulb7rc7vhujp2hrho12hs/1612324875000/00484516897554883881/04318519678129211259/15dCNcmKskcFVjs7R0ElQkR61Ex53uJpM?e=download&authuser=0&nonce=bdplkjva1bpi4&user=04318519678129211259&hash=ektoa9kgh4ueg2iauui97s4ih2jaqnlg" -c -O 'data.csv'
```

```
--2021-02-03 04:02:47--  https://doc-0o-58-docs.googleusercontent.com/docs/securesc/aa6ktludmnluspcle3g
r7p17lfha05rn/utonp9io8o9ulb7rc7vhujp2hrho12hs/1612324875000/00484516897554883881/04318519678129211259/
15dCNcmKskcFVjs7R0ElQkR61Ex53uJpM?e=download&authuser=0&nonce=bdplkjva1bpi4&user=04318519678129211259&h
ash=ektoa9kgh4ueg2iauui97s4ih2jaqnlg
Resolving doc-0o-58-docs.googleusercontent.com (doc-0o-58-docs.googleusercontent.com)... 74.125.134.132
, 2607:f8b0:400c:c00::84
Connecting to doc-0o-58-docs.googleusercontent.com (doc-0o-58-docs.googleusercontent.com)|74.125.134.13
2|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 886913 (866K) [text/csv]
Saving to: 'data.csv'

data.csv            100%[===================>] 866.13K  --.-KB/s    in 0.009s

2021-02-03 04:02:47 (98.7 MB/s) - 'data.csv' saved [886913/886913]
```

In [3]:

```
data = pd.read_csv("/content/data.csv")
```

In [4]:

```
data.head()
```

Out[4]:

	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 [5]:

```
# standadize the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(data['f1'].values.reshape(-1, 1))
data['f1']=scaler.transform(data['f1'].values.reshape(-1, 1))
scaler = StandardScaler()
scaler.fit(data['f2'].values.reshape(-1, 1))
data['f2']=scaler.transform(data['f2'].values.reshape(-1, 1))
```

In [6]:

```
data.head()
```

Out[6]:

	f1	f2	label
0	0.670394	1.593406	0.0
1	0.126651	1.435372	0.0
2	0.173875	1.441062	1.0
3	1.462492	-0.562725	0.0
4	-1.074251	1.417835	0.0

In [7]:

```
# balanced data
```

```
data['label'].value_counts()
```

Out[7]:

```
1.0    10000
0.0    10000
Name: label, dtype: int64
```

In [8]:

```
X = data[['f1','f2']]
y = data['label']
```

In [9]:

```
X_train,X_test,y_train,y_test  = train_test_split(X,y,test_size=0.33,random_state=42)
```

In [10]:

```
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

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

Callbacks

In [11]:

```
# micro_f1 score call back function

class f1_auc(tf.keras.callbacks.Callback):
  def __init__(self, train_data,validation_data,logs={}):
        self.train_data      = train_data
        self.validation_data = validation_data
  def set_model(self, model):
        self.model = model

  def on_epoch_end(self,epoch, logs={}):

    # F1_micro score and auc score on validation data.
    predict_f1  =  (np.asarray(self.model.predict(self.validation_data[0]))).round()
    predict_auc =   np.asarray(self.model.predict(self.validation_data[0]))
    target      =   self.validation_data[1]

    val_auc     =   roc_auc_score(target,predict_auc)
    f1          =   f1_score(target, predict_f1)

    # auc score on train_data
    predict_auc =  np.asarray(self.model.predict(self.train_data[0]))
    target      =  self.train_data[1]
    train_auc   =  roc_auc_score(target,predict_auc)

    print( 92*"-" + "\n"+"| f1_micro: {}, train_auc: {}, val_auc: {} ".format(f1,train_auc,val_auc) +"|
\n" + 92*"-")
```

In [12]:

```
# decay Learning  call back function
class LearnigRate(tf.keras.callbacks.Callback):
  def on_train_begin(self, logs={}):
    self.history ={'val_accuracy':1,'epoch_number':0,'count':0}
  def on_epoch_end(self,epoch,logs={}):
    self.history['epoch_number']+=1
    self.history['count']+=1

    # first epoch accuracy is updated into history to compare it from next.
```

```
    if (self.history['epoch_number']==1):
      self.history['val_accuracy'] = logs['val_accuracy']

    if (logs['val_accuracy'] < self.history['val_accuracy']):
      # decreasing 10 percent
      self.model.optimizer.lr = self.model.optimizer.lr*(1-0.1)
      self.history['val_accuracy'] = logs['val_accuracy']
    if self.history['count'] == 3:
      # decreasing 5 percent
      self.model.optimizer.lr = self.model.optimizer.lr*(1-0.05)
      self.history['count']=0

printLog = LearnigRate()
```

In [13]:

```
# https://stackoverflow.com/questions/53400472/keras-model-weights-for-some-layers-become-all-nans

class TerminateNaN(tf.keras.callbacks.Callback):
  def on_epoch_end(self,epoch,logs={}):
    loss = logs['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
    for layer in self.model.layers:
      # in each layer get_weights give a list of two np.array ( weights and bias of nueron)
      weights = layer.get_weights()
      # checking each array in weights has nan or inf by summing values first.
      # because sum of a array containing atleast one nan values gives nan as result.. similarly for in
f.
      for each_array in (weights):
        val = np.sum(each_array)
        if np.isinf(val) or np.isnan(val):
          print("weights has nan values and terminated at epoch {}".format(epoch))
          self.model.stop_training = True

terminateNan = TerminateNaN()
```

Model-1

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.
4. Analyze your output and training process.

In [14]:

```
# create directory CB (call back) and inside create save_model1 ( for first model weights)
destination_1 = "/content/CB"
destination_2= "/content/CB/save_model1"

if not os.path.isdir(destination_1):
```

```
    os.makedirs(destination_1)

# create save_mode2 directory for model-2
if not os.path.isdir(destination_2):
    os.makedirs(destination_2)

# !rm -rf /content/CB/save_model1/weights*
```

In [15]:

```
tf.keras.backend.clear_session()

initializer = tf.keras.initializers.RandomUniform(0,1)
# Input Layer
input_layer = Input(shape=(2,),name="input")
# Dense hidden layer
layer1 = Dense(4,activation='tanh',kernel_initializer=initializer,name="hidden1")(input_layer)
layer2 = Dense(4,activation='tanh',kernel_initializer=initializer,name='hidden2')(layer1)
layer3 = Dense(4,activation='tanh',kernel_initializer=initializer,name="hidden3")(layer2)
layer4 = Dense(4,activation='tanh',kernel_initializer=initializer,name="hidden4")(layer3)
layer5 = Dense(4,activation='tanh',kernel_initializer=initializer,name="hidden5")(layer4)


output = Dense(1,activation='sigmoid',kernel_initializer=initializer,name="output")(layer5)
model = Model(inputs=input_layer,outputs=output)

optimizer = tf.keras.optimizers.SGD(learning_rate=0.001,momentum=0.9)
earlyStopping= tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2, verbose=1,mode='max
')

# clear weights from previous !
!rm -rf ./CB/save_model1/weights*
filepath="/content/CB/save_model1/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, save_weights_only=True,monitor='val_accuracy',mode='max
',save_best_only=True)

# Clear any logs from previous !
!rm -rf ./logs1*
log_dir = "logs1/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

model.compile(optimizer=optimizer,loss="binary_crossentropy",metrics=['accuracy'])
train_data      = [X_train,y_train]
validation_data = [X_test,y_test]
model.fit(X_train,y_train,epochs=15,validation_data=(X_test,y_test),batch_size=50,callbacks=[tensorboar
d_callback,earlyStopping,terminateNan,checkpoint,printLog,f1_auc(train_data,validation_data)])
tf.keras.backend.clear_session()
```

```
Epoch 1/15
  3/268 [..............................] - ETA: 7s - loss: 0.9756 - accuracy: 0.4789  WARNING:tensorflo
w:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0032s vs `on_t
rain_batch_end` time: 0.0085s). Check your callbacks.
268/268 [==============================] - 2s 4ms/step - loss: 0.8433 - accuracy: 0.5048 - val_loss: 0.
6999 - val_accuracy: 0.5015
-------------------------------------------------------------------------------------------------
| f1_micro: 0.505411906193626, train_auc: 0.4980259924838326, val_auc: 0.5002463291673936 |
-------------------------------------------------------------------------------------------------
Epoch 2/15
268/268 [==============================] - 0s 2ms/step - loss: 0.6958 - accuracy: 0.5034 - val_loss: 0.
6934 - val_accuracy: 0.5030
-------------------------------------------------------------------------------------------------
| f1_micro: 0.5094226742446903, train_auc: 0.5317893445964459, val_auc: 0.5277295934765242 |
-------------------------------------------------------------------------------------------------
Epoch 3/15
268/268 [==============================] - 1s 2ms/step - loss: 0.6933 - accuracy: 0.5045 - val_loss: 0.
6932 - val_accuracy: 0.5017
-------------------------------------------------------------------------------------------------
| f1_micro: 0.5132455231611661, train_auc: 0.5413658974513198, val_auc: 0.5365457901358139 |
-------------------------------------------------------------------------------------------------
Epoch 4/15
268/268 [==============================] - 1s 2ms/step - loss: 0.6931 - accuracy: 0.5050 - val_loss: 0.
6931 - val_accuracy: 0.5017
-------------------------------------------------------------------------------------------------
| f1_micro: 0.5192223359158018, train_auc: 0.5504311548923032, val_auc: 0.5432218844984802 |
-------------------------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------------------------------
Epoch 00004: early stopping
```

In [ ]:

```
!tensorboard dev upload --logdir ./logs1 \
  --name "Call backs Model:1" \
  --description " from ReSumbit_Call_Backs_Assignment.ipynb " \
  --one_shot
```

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 [20]:

```
# create save_mode2 directory for model-2
destination = "/content/CB/save_model2"
if not os.path.isdir(destination):
    os.makedirs(destination)
```

In [21]:

```
## using relu
tf.keras.backend.clear_session()
initializer = tf.keras.initializers.RandomUniform(0,1)
# Input Layer
input_layer = Input(shape=(2,))
# Dense hidden layer
layer1 = Dense(8,activation='relu',kernel_initializer=initializer)(input_layer)
layer2 = Dense(8,activation='relu',kernel_initializer=initializer)(layer1)
layer3 = Dense(8,activation='relu',kernel_initializer=initializer)(layer2)
layer4 = Dense(8,activation='relu',kernel_initializer=initializer)(layer3)
layer5 = Dense(4,activation='relu',kernel_initializer=initializer)(layer4)



output = Dense(1,activation='sigmoid',kernel_initializer=initializer)(layer5)
model = Model(inputs=input_layer,outputs=output)

optimizer = tf.keras.optimizers.SGD(learning_rate=0.001,momentum=0.9)

earlyStopping= tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2, verbose=1,mode='max
')

# clear weights from previous !
# !rm -rf ./CB/save_model2/weights*
filepath="/content/CB/save_model2/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, save_weights_only=True,monitor='val_accuracy',mode='max
',save_best_only=True)

# Clear any logs from previous !
# !rm -rf ./logs2*
log_dir = "logs2/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

model.compile(optimizer=optimizer,loss=tf.keras.losses.BinaryCrossentropy(),metrics=['accuracy'])
train_data      = [X_train,y_train]
validation_data = [X_test,y_test]
model.fit(X_train,y_train,epochs=10,validation_data=(X_test,y_test),batch_size=16,callbacks=[tensorboar
d_callback,earlyStopping,terminateNan,checkpoint,printLog,f1_auc(train_data,validation_data)])
tf.keras.backend.clear_session()
```

```
Epoch 1/10
```

```
  3/838 [..............................] - ETA: 32s - loss: 61.4879 - accuracy: 0.5660 WARNING:tensorfl
ow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0029s vs `on_
train_batch_end` time: 0.0128s). Check your callbacks.
838/838 [==============================] - 2s 2ms/step - loss: 2.6934 - accuracy: 0.5034 - val_loss: 0.
6931 - val_accuracy: 0.4988
-----------------------------------------------------------------------------------------------------
| f1_micro: 0.665453074433657, train_auc: 0.5, val_auc: 0.5001510574018126 |
-----------------------------------------------------------------------------------------------------
Epoch 2/10
838/838 [==============================] - 2s 2ms/step - loss: 0.6932 - accuracy: 0.5004 - val_loss: 0.
6931 - val_accuracy: 0.4988
-----------------------------------------------------------------------------------------------------
| f1_micro: 0.665453074433657, train_auc: 0.5, val_auc: 0.5001510574018126 |
-----------------------------------------------------------------------------------------------------
Epoch 3/10
838/838 [==============================] - 1s 2ms/step - loss: 0.6932 - accuracy: 0.5005 - val_loss: 0.
6931 - val_accuracy: 0.5015
-----------------------------------------------------------------------------------------------------
| f1_micro: 0.0, train_auc: 0.5, val_auc: 0.5001510574018126 |
-----------------------------------------------------------------------------------------------------
Epoch 4/10
838/838 [==============================] - 2s 2ms/step - loss: 0.6932 - accuracy: 0.4979 - val_loss: 0.
6931 - val_accuracy: 0.5014
-----------------------------------------------------------------------------------------------------
| f1_micro: 0.0, train_auc: 0.5, val_auc: 0.5001510574018126 |
-----------------------------------------------------------------------------------------------------
Epoch 5/10
838/838 [==============================] - 2s 2ms/step - loss: 0.6932 - accuracy: 0.5081 - val_loss: 0.
6932 - val_accuracy: 0.4988
-----------------------------------------------------------------------------------------------------
| f1_micro: 0.665453074433657, train_auc: 0.5, val_auc: 0.5001510574018126 |
-----------------------------------------------------------------------------------------------------
Epoch 00005: early stopping
```

In [ ]:

```
!tensorboard dev upload --logdir ./logs2 \
  --name "Call backs Model:2" \
  --description " from ReSumbit_Call_Backs_Assignment.ipynb " \
  --one_shot
```

Model-3

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 [24]:

```
# create save_mode2 directory for model-2
destination = "/content/CB/save_model3"
if not os.path.isdir(destination):
    os.makedirs(destination)
```

In [25]:

```
## using relu
tf.keras.backend.clear_session()
initializer = tf.keras.initializers.he_uniform()
# Input Layer
input_layer = Input(shape=(2,))
# Dense hidden layer
layer1 = Dense(4,activation='relu',kernel_initializer=initializer)(input_layer)
layer2 = Dense(4,activation='relu',kernel_initializer=initializer)(layer1)
```

```
layer3 = Dense(4,activation='relu',kernel_initializer=initializer)(layer2)
layer4 = Dense(4,activation='relu',kernel_initializer=initializer)(layer3)
layer5 = Dense(4,activation='relu',kernel_initializer=initializer)(layer4)



output = Dense(1,activation='sigmoid',kernel_initializer=initializer)(layer5)
model = Model(inputs=input_layer,outputs=output)

optimizer = tf.keras.optimizers.SGD(learning_rate=0.02,momentum=0.9)

earlyStopping= tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2, verbose=1,mode='aut
o')

# clear weights from previous !
# !rm -rf ./CB/save_model3/weights*
filepath="/content/CB/save_model3/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, save_weights_only=True,monitor='val_accuracy',mode='max
',save_best_only=True)

# Clear any logs from previous !
# !rm -rf ./logs3*
log_dir = "logs3/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

model.compile(optimizer=optimizer,loss=tf.keras.losses.BinaryCrossentropy(),metrics=['accuracy'])
train_data      = [X_train,y_train]
validation_data = [X_test,y_test]
model.fit(X_train,y_train,epochs=15,validation_data=(X_test,y_test),batch_size=16,callbacks=[tensorboar
d_callback,earlyStopping,terminateNan,checkpoint,printLog,f1_auc(train_data,validation_data)])
tf.keras.backend.clear_session()
```

```
Epoch 1/15
  3/838 [..............................] - ETA: 39s - loss: 0.6933 - accuracy: 0.4132 WARNING:tensorflo
w:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0022s vs `on_t
rain_batch_end` time: 0.0151s). Check your callbacks.
838/838 [==============================] - 2s 2ms/step - loss: 0.6811 - accuracy: 0.5511 - val_loss: 0.
6690 - val_accuracy: 0.5895
--------------------------------------------------------------------------------------------------
| f1_micro: 0.6845231163386515, train_auc: 0.7100540990289576, val_auc: 0.6981318010266395 |
--------------------------------------------------------------------------------------------------
Epoch 2/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6275 - accuracy: 0.6418 - val_loss: 0.
6324 - val_accuracy: 0.6491
--------------------------------------------------------------------------------------------------
| f1_micro: 0.6138712904301433, train_auc: 0.725795947418016, val_auc: 0.7086954884801513 |
--------------------------------------------------------------------------------------------------
Epoch 3/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6196 - accuracy: 0.6549 - val_loss: 0.
6188 - val_accuracy: 0.6561
--------------------------------------------------------------------------------------------------
| f1_micro: 0.6268902038132808, train_auc: 0.7361471845559915, val_auc: 0.7191143169358765 |
--------------------------------------------------------------------------------------------------
Epoch 4/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6085 - accuracy: 0.6660 - val_loss: 0.
6198 - val_accuracy: 0.6518
--------------------------------------------------------------------------------------------------
| f1_micro: 0.6103763987792472, train_auc: 0.7372634156012822, val_auc: 0.7203764038237265 |
--------------------------------------------------------------------------------------------------
Epoch 5/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6110 - accuracy: 0.6620 - val_loss: 0.
6168 - val_accuracy: 0.6579
--------------------------------------------------------------------------------------------------
| f1_micro: 0.63872, train_auc: 0.740096057242275, val_auc: 0.721671916179212 |
--------------------------------------------------------------------------------------------------
Epoch 6/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6127 - accuracy: 0.6681 - val_loss: 0.
6232 - val_accuracy: 0.6576
--------------------------------------------------------------------------------------------------
| f1_micro: 0.6945120302784535, train_auc: 0.7428372306465374, val_auc: 0.7240486597673073 |
--------------------------------------------------------------------------------------------------
Epoch 7/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6044 - accuracy: 0.6704 - val_loss: 0.
6187 - val_accuracy: 0.6632
--------------------------------------------------------------------------------------------------
| f1_micro: 0.6670660476261795, train_auc: 0.7394792481159459, val_auc: 0.7233450720392291 |
```

```
| f1_micro: 0.6670660476261795, train_auc: 0.7394792481159459, val_auc: 0.7233450720392291 |
-------------------------------------------------------------------------------------------------
Epoch 8/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6083 - accuracy: 0.6660 - val_loss: 0.
6420 - val_accuracy: 0.6445
-------------------------------------------------------------------------------------------------
| f1_micro: 0.5753077480086894, train_auc: 0.7282875992149681, val_auc: 0.7128369865655332 |
-------------------------------------------------------------------------------------------------
Epoch 9/15
838/838 [==============================] - 2s 2ms/step - loss: 0.6148 - accuracy: 0.6590 - val_loss: 0.
6212 - val_accuracy: 0.6506
-------------------------------------------------------------------------------------------------
| f1_micro: 0.5893874643874644, train_auc: 0.7419196968583134, val_auc: 0.7249000449958218 |
-------------------------------------------------------------------------------------------------
Epoch 00009: early stopping
```

In [ ]:

```
!tensorboard dev upload --logdir ./logs3 \
  --name "Call backs Model:3" \
  --description " from ReSumbit_Call_Backs_Assignment.ipynb " \
  --one_shot
```

Model-4

Model-3

1. Using swish 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 [33]:

```
# create save_mode2 directory for model-2
destination = "/content/CB/save_model4"
if not os.path.isdir(destination):
    os.makedirs(destination)
```

In [41]:

```
## using swish --> reference: https://arxiv.org/pdf/1710.05941v1.pdf
tf.keras.backend.clear_session()
initializer = tf.keras.initializers.he_uniform()
# Input Layer
input_layer = Input(shape=(2,))
# Dense hidden layer
# Dense hidden layer
layer1 = Dense(4,activation='swish',kernel_initializer=initializer)(input_layer)
layer2 = Dense(4,activation='swish',kernel_initializer=initializer)(layer1)
layer3 = Dense(4,activation='swish',kernel_initializer=initializer)(layer2)
layer4 = Dense(4,activation='swish',kernel_initializer=initializer)(layer3)
layer5 = Dense(4,activation='swish',kernel_initializer=initializer)(layer4)



output = Dense(1,activation='sigmoid',kernel_initializer=initializer)(layer5)
model = Model(inputs=input_layer,outputs=output)


optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)

earlyStopping= tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2, verbose=1,mode='aut
o')

# clear weights from previous !
!rm -rf ./CB/save_model4/weights*
```

```
filepath="/content/CB/save_model4/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, save_weights_only=True,monitor='val_accuracy',mode='max
',save_best_only=True)

# Clear any logs from previous !
!rm -rf ./logs4*
log_dir = "logs4/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


model.compile(optimizer=optimizer,loss=tf.keras.losses.BinaryCrossentropy(),metrics=['accuracy'])
train_data      = [X_train,y_train]
validation_data = [X_test,y_test]
model.fit(X_train,y_train,epochs=20,validation_data=(X_test,y_test),batch_size=16,callbacks=[tensorboar
d_callback,earlyStopping,terminateNan,checkpoint,printLog,f1_auc(train_data,validation_data)])
tf.keras.backend.clear_session()
```

```
Epoch 1/20
  3/838 [..............................] - ETA: 39s - loss: 0.6789 - accuracy: 0.5417 WARNING:tensorflo
w:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0020s vs `on_t
rain_batch_end` time: 0.0153s). Check your callbacks.
838/838 [==============================] - 2s 2ms/step - loss: 0.6486 - accuracy: 0.6189 - val_loss: 0.
6245 - val_accuracy: 0.6538
------------------------------------------------------------------------------------------------
| f1_micro: 0.6134325833192353, train_auc: 0.7387138532275634, val_auc: 0.7205110698904489 |
------------------------------------------------------------------------------------------------
Epoch 2/20
838/838 [==============================] - 1s 2ms/step - loss: 0.6097 - accuracy: 0.6682 - val_loss: 0.
6131 - val_accuracy: 0.6661
------------------------------------------------------------------------------------------------
| f1_micro: 0.6682721252257676, train_auc: 0.7404143916560296, val_auc: 0.7239086676645332 |
------------------------------------------------------------------------------------------------
Epoch 3/20
838/838 [==============================] - 1s 2ms/step - loss: 0.6074 - accuracy: 0.6684 - val_loss: 0.
6188 - val_accuracy: 0.6539
------------------------------------------------------------------------------------------------
| f1_micro: 0.6233509234828497, train_auc: 0.739928246665731, val_auc: 0.722781797812652 |
------------------------------------------------------------------------------------------------
Epoch 4/20
838/838 [==============================] - 1s 2ms/step - loss: 0.5986 - accuracy: 0.6791 - val_loss: 0.
6146 - val_accuracy: 0.6564
------------------------------------------------------------------------------------------------
| f1_micro: 0.6221259580139953, train_auc: 0.7428372640616263, val_auc: 0.7255648812202132 |
------------------------------------------------------------------------------------------------
Epoch 00004: early stopping
```

In [42]:

```
!tensorboard dev upload --logdir ./logs4 \
  --name "Call backs Model:4" \
  --description " ReSumbit_Call_Backs_Assignment.ipynb " \
  --one_shot
```

```
2021-02-03 05:07:01.161696: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfull
y opened dynamic library libcudart.so.10.1

New experiment created. View your TensorBoard at: https://tensorboard.dev/experiment/L62E5OMjRBamxhlKgG
lW2Q/

[2021-02-03T05:07:03] Started scanning logdir.
[2021-02-03T05:07:04] Total uploaded: 16 scalars, 48 tensors (8.4 kB), 1 binary objects (59.3 kB)
[2021-02-03T05:07:04] Done scanning logdir.


Done. View your TensorBoard at https://tensorboard.dev/experiment/L62E5OMjRBamxhlKgGlW2Q/
```

In [ ]:

```
# !rm -r ./CB
# !rm -r ./logs*
```

In [ ]:

```
```