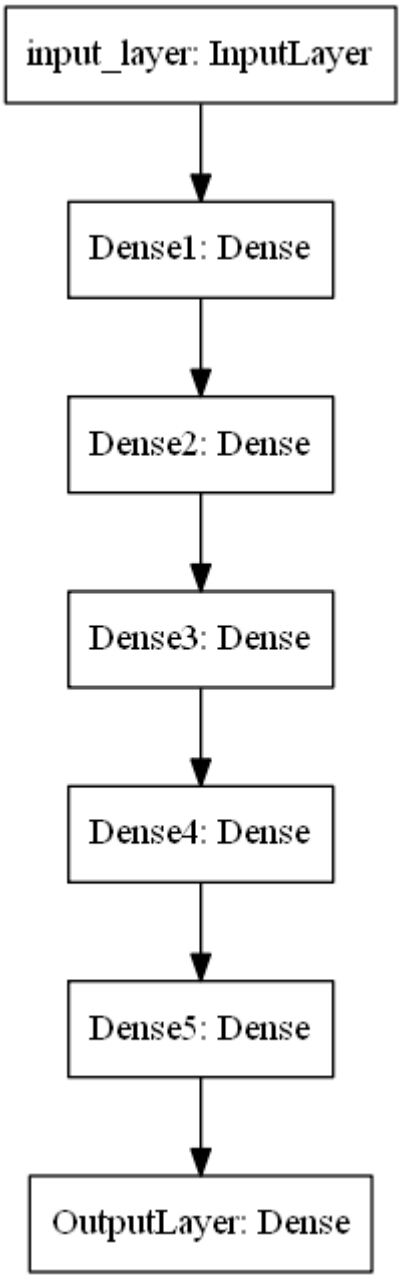
1. Download the data from here

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 aft

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

5. you have to decay learning based on below conditions
   Cond1. If your validation accuracy at that epoch is less than previous epoch accura
   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 te

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

8. Use tensorboard for every model and analyse your gradients. (you need to upload the scre

9. use cross entropy as loss function

10. Try the architecture params as given below.

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

Choose Files No file chosen

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
from sklearn.metrics import f1_score
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/compat/
Instructions for updating:
non-resource variables are not supported in the long term
```

```
import numpy as np
import pandas as pd
#import tensorflow as tf
#enabled to get instant output. if you don't need, you can use session concept which was dicu
#tf.enable_eager_execution()
df = pd.read_csv('data.csv')
print(df.head())
X =df[['f1','f2']]
y = df['label']
```

```
         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
```

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

    def on_train_begin(self, logs={}):
        ## on begin of training, we are creating a instance varible called history
```

```
            ## it is a dict with keys [loss, acc, val_loss, val_acc]
            self.history={'loss': [],'acc': [],'val_loss': [],'val_acc': []}

        def on_epoch_end(self, epoch, logs={}):
            ## on end of each epoch, we will get logs and update the self.history dict
            self.history['loss'].append(logs.get('loss'))
            self.history['acc'].append(logs.get('acc'))
            if logs.get('val_loss', -1) != -1:
                self.history['val_loss'].append(logs.get('val_loss'))
            if logs.get('val_acc', -1) != -1:
                self.history['val_acc'].append(logs.get('val_acc'))
            #self.lr.append(step_decay(len(self.history['loss'])))

    history_own=LossHistory()


    import math
    from tensorflow.keras.callbacks import LearningRateScheduler
    # Decrease LR based on condtion 2 -- decrease by 5% every 3 epochs
    def step_decay(epoch,lr):
      if (epoch+1)%3==0:
        lr = 0.95*lr
      return lr
    lrate = LearningRateScheduler(step_decay)


    # Terminate training if either weights or loss value has Nan
    class TerminateNan(tf.keras.callbacks.Callback):
      def on_epoch_end(self,epoch,log={}):
        loss = logs.get('loss')
        if loss not in None:
          if np.isnan(loss) or np.isinf(loss):
            print('Invalid loss and terminated at epoch {}'.format(epoch))
            self.model.stop_training = True


    #Custom callback to implment f1score for at end of each epoch
    class Metrics(tf.keras.callbacks.Callback):
      def __init__(self,validation):
        super(Metrics,self).__init__()
        self.validation = validation
      def on_train_begin(self,logs={}):
        self.val_f1s = []
      def on_epoch_end(self,epoch,logs={}):
        val_predict = (np.asarray(self.model.predict(self.validation[0]))).round()
        val_target = self.validation[1]
        val_f1_score = f1_score(val_target,val_predict,average='micro')

        self.val_f1s.append(round(val_f1_score,6))
        print('--f1_score: {0}'.format(round(val_f1_score,2)))


    from sklearn.metrics import roc auc score
```

```
#Implementing AUC score as a callback
class AUC_Callback(tf.keras.callbacks.Callback):
  def __init__(self,validation):
    super(AUC_Callback,self).__init__()
    self.validation = validation
  def on_train_begin(self,logs={}):
    self.val_auc = []
  def on_epoch_end(self,epoch,logs={}):
    #print(self.model)
    val_predict =(self.model.predict(self.validation[0]))
    #print(val_predict[10:20])
    val_target = self.validation[1]
    val_auc = roc_auc_score(val_target,val_predict)

    self.val_auc.append(round(val_auc,6))
    print('--AUC: {0}'.format(round(val_auc,2)))


from sklearn.model_selection import train_test_split
#Standardizing the inputs
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform (X_test)


term = TerminateNan()
from tensorflow.keras.layers import Dense,Input,Activation,Dropout
from tensorflow.keras.models import Model


from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import EarlyStopping
earlystop = EarlyStopping(monitor='val_acc', patience=2, verbose=1)


from tensorflow.keras.callbacks import ReduceLROnPlateau
reduce = ReduceLROnPlateau(monitor='val_acc',factor=0.9,patience=1,verbose=1)
# To reduce learning rate based on condition 1 --- reduce lr by 10% if 'val_acc' is not impro



# !pip install -q tf-nightly-2.0-preview
# if you want to use the tf2.0 please uncomment the above line
# Load the TensorBoard notebook extension

# there are other ways of doing this: https://www.dlology.com/blog/quick-guide-to-run-tensorb
%load_ext tensorboard


# Clear any logs from previous runs
!rm -rf ./logs/
```

```
import datetime
log_dir = "logs/fit" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write
```

Model 1

```
auc = AUC_Callback(validation=(X_test,y_test))
input_layer = Input(shape=(2,))
dense1 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense2 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense3 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense4 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense5 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.RandomUniform(min
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.RandomUniform(
optimizer  = tf.keras.optimizers.SGD(learning_rate=0.1,momentum=0.9)
model = Model(inputs= input_layer,outputs=output)
model.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])
callback_list=[Metrics(validation=(X_test,y_test)),auc,earlystop,history_own,lrate,reduce,ten
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=128,callbacks=
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/i
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the construct
Train on 16000 samples, validate on 4000 samples
Epoch 1/10
  128/16000 [..............................] - ETA: 12s - loss: 0.9445 - acc: 0.5781WARN
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch ti
 2176/16000 [===>..........................] - ETA: 0s - loss: 0.8071 - acc: 0.5028 /usr
  warnings.warn('`Model.state_updates` will be removed in a future version. '
13696/16000 [========================>.....] - ETA: 0s - loss: 0.7159 - acc: 0.5004--f1_
--AUC: 0.49
16000/16000 [==============================] - 1s 47us/sample - loss: 0.7130 - acc: 0.50
Epoch 2/10
11264/16000 [====================>.........] - ETA: 0s - loss: 0.6945 - acc: 0.4975--f1_
--AUC: 0.52

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6947 - acc: 0.49
Epoch 3/10
12544/16000 [======================>.......] - ETA: 0s - loss: 0.6941 - acc: 0.4971--f1_
--AUC: 0.5
16000/16000 [==============================] - 0s 21us/sample - loss: 0.6947 - acc: 0.49
Epoch 4/10
12288/16000 [======================>.......] - ETA: 0s - loss: 0.6951 - acc: 0.4927--f1_
--AUC: 0.49

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.07695000171661377.
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6950 - acc: 0.49
Epoch 5/10
13312/16000 [=======================>......] - ETA: 0s - loss: 0.6941 - acc: 0.5013--f1_
```

```
--AUC: 0.5

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.06925499886274337.
16000/16000 [==============================] - 0s 20us/sample - loss: 0.6940 - acc: 0.50
Epoch 00005: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f5bb6b38be0>
```

Model 2

```
import datetime

log_dir = "logs/fit" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback2 = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, writ


auc = AUC_Callback(validation=(X_test,y_test))
input_layer = Input(shape=(2,))
dense1 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense2 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense3 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense4 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(min
dense5 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.RandomUniform(min
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.RandomUniform(
optimizer  = tf.keras.optimizers.SGD(learning_rate=0.1,momentum=0.9)
model = Model(inputs= input_layer,outputs=output)
model.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])
callback_list=[Metrics(validation=(X_test,y_test)),auc,earlystop,history_own,lrate,reduce,ten
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=128,callbacks=

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/i
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the construct
Train on 16000 samples, validate on 4000 samples
Epoch 1/10
  128/16000 [..............................] - ETA: 13s - loss: 18.0577 - acc: 0.4688WAR
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch ti
 2176/16000 [===>..........................] - ETA: 0s - loss: 1.7174 - acc: 0.5009  /us
  warnings.warn('`Model.state_updates` will be removed in a future version. '
13696/16000 [========================>.....] - ETA: 0s - loss: 0.8561 - acc: 0.5014--f1_
--AUC: 0.5
16000/16000 [==============================] - 1s 47us/sample - loss: 0.8326 - acc: 0.50
Epoch 2/10
11904/16000 [=====================>........] - ETA: 0s - loss: 0.6937 - acc: 0.5060--f1_
--AUC: 0.5

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6938 - acc: 0.50
Epoch 3/10
15104/16000 [===========================>..] - ETA: 0s - loss: 0.6939 - acc: 0.4976--f1_
```

```
--AUC: 0.5

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.07695000171661377.
16000/16000 [==============================] - 0s 24us/sample - loss: 0.6938 - acc: 0.49
Epoch 00003: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f1c1eb4ebe0>
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 5)                 15
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 6
=================================================================
Total params: 141
Trainable params: 141
Non-trainable params: 0
_________________________________________________________________
```

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.

Model 3

```
import datetime
log_dir = "logs/fit" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write
```

```
auc = AUC_Callback(validation=(X_test,y_test))
input_layer = Input(shape=(2,))
dense1 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(inpu
dense2 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
dense3 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
dense4 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
dense5 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.he_normal())(d
optimizer  = tf.keras.optimizers.SGD(learning_rate=0.1,momentum=0.9)
model = Model(inputs= input_layer,outputs=output)
model.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])
callback_list=[Metrics(validation=(X_test,y_test)),auc,earlystop,history_own,lrate,reduce,ten
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=128,callbacks=
```

```
Train on 16000 samples, validate on 4000 samples
Epoch 1/10
```

```
  128/16000 [..............................] - ETA: 13s - loss: 0.8072 - acc: 0.5781WARN
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch ti
 2816/16000 [====>.........................] - ETA: 0s - loss: 0.7034 - acc: 0.5156 /usr
  warnings.warn('`Model.state_updates` will be removed in a future version. '
14336/16000 [=========================>....] - ETA: 0s - loss: 0.6792 - acc: 0.5670--f1_
--AUC: 0.71
16000/16000 [==============================] - 1s 46us/sample - loss: 0.6742 - acc: 0.57
Epoch 2/10
11520/16000 [====================>.........] - ETA: 0s - loss: 0.6111 - acc: 0.6638--f1_
--AUC: 0.71
16000/16000 [==============================] - 0s 23us/sample - loss: 0.6165 - acc: 0.66
Epoch 3/10
11520/16000 [====================>.........] - ETA: 0s - loss: 0.6115 - acc: 0.6616--f1_
--AUC: 0.7

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6098 - acc: 0.66
Epoch 4/10
11520/16000 [====================>.........] - ETA: 0s - loss: 0.6084 - acc: 0.6688--f1_
--AUC: 0.71
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6070 - acc: 0.66
Epoch 5/10
12928/16000 [=========================>.....] - ETA: 0s - loss: 0.6088 - acc: 0.6675--f1_
--AUC: 0.71
16000/16000 [==============================] - 0s 21us/sample - loss: 0.6062 - acc: 0.67
Epoch 6/10
12544/16000 [======================>.......] - ETA: 0s - loss: 0.6042 - acc: 0.6683--f1_
--AUC: 0.71

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.07310250028967857.
16000/16000 [==============================] - 0s 23us/sample - loss: 0.6055 - acc: 0.66
Epoch 7/10
11136/16000 [===================>..........] - ETA: 0s - loss: 0.6079 - acc: 0.6635--f1_
--AUC: 0.71

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.06579225361347199.
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6085 - acc: 0.66
Epoch 00007: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f9f444ebe48>
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 5)                 15
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 30
```

```
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 6
=================================================================
Total params: 141
Trainable params: 141
Non-trainable params: 0
_________________________________________________________________
```

Model 4

```
import datetime
log_dir = "logs/fit" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write
```

```
#Adding dropout and using adam optimizer
auc = AUC_Callback(validation=(X_test,y_test))
input_layer = Input(shape=(2,))
dense1 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(inpu
dense2 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
dense3 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
dense4 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
dense5 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_normal())(dens
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.he_normal())(d
optimizer  = tf.keras.optimizers.Adam(learning_rate=0.01)
model = Model(inputs= input_layer,outputs=output)
model.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])
callback_list=[Metrics(validation=(X_test,y_test)),auc,earlystop,history_own,lrate,reduce,ten
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=11,batch_size=128,callbacks=
```

```
Train on 16000 samples, validate on 4000 samples
Epoch 1/11
  128/16000 [..............................] - ETA: 14s - loss: 0.8995 - acc: 0.4609WARN
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch ti
 2432/16000 [===>..........................] - ETA: 0s - loss: 0.7156 - acc: 0.4955 /usr
  warnings.warn('`Model.state_updates` will be removed in a future version. '
13824/16000 [========================>.....] - ETA: 0s - loss: 0.6885 - acc: 0.5407--f1_
--AUC: 0.63
16000/16000 [==============================] - 1s 49us/sample - loss: 0.6846 - acc: 0.54
Epoch 2/11
11904/16000 [=====================>........] - ETA: 0s - loss: 0.6387 - acc: 0.6394--f1_
--AUC: 0.71
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6339 - acc: 0.64
Epoch 3/11
11264/16000 [====================>.........] - ETA: 0s - loss: 0.6070 - acc: 0.6662--f1_
--AUC: 0.71
16000/16000 [==============================] - 0s 23us/sample - loss: 0.6083 - acc: 0.66
Epoch 4/11
11392/16000 [====================>.........] - ETA: 0s - loss: 0.6037 - acc: 0.6631--f1_
```

```
--AUC: 0.71

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.008549999725073577.
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6033 - acc: 0.66
Epoch 5/11
11648/16000 [====================>.........] - ETA: 0s - loss: 0.6016 - acc: 0.6673--f1_
--AUC: 0.71

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.007694999501109123.
16000/16000 [==============================] - 0s 22us/sample - loss: 0.6034 - acc: 0.66
Epoch 00005: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f9079b8aa20>
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 5)                 15
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 6
=================================================================
Total params: 141
Trainable params: 141
Non-trainable params: 0
_________________________________________________________________
```