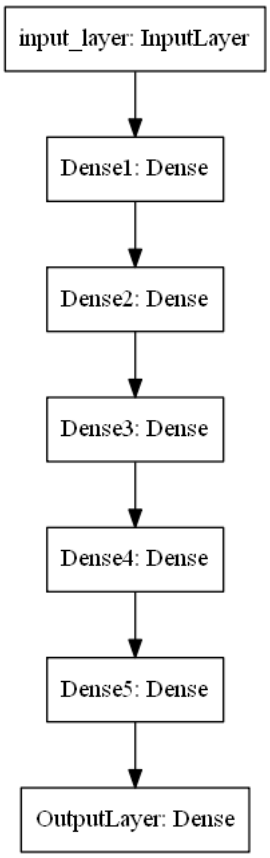
1. Download the data from here. You have to use data.csv file for this assignment
2. Code the model to classify data like below image. You can use any number of units in your Dense layers.



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 accuracy, you have to decrese the
       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)

Note

Make sure that you are plotting tensorboard plots either in your notebook or you can try to create a pdf file with all the tensorboard screenshots.Please write your analysis of tensorboard results for each model.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#Import Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
import keras
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import LearningRateScheduler
import random as rn
from tensorflow import keras
import datetime, os

from keras.callbacks import Callback
from sklearn.metrics import roc_auc_score, f1_score
```

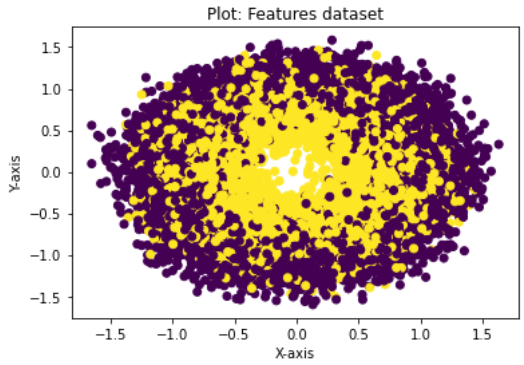
Task-1: Loading Dataset

```
dataset=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/AAIC_Assignments/solving/20_Working-with-Callbacks/data.csv")
dataset.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

```
#Independent variables
X=dataset[["f1","f2"]].values
#Class variable
y=dataset['label'].values
```

```
#Plotting the dataset
import matplotlib.pyplot as plt
plt.scatter(dataset['f1'], dataset['f2'], c=y)
plt.title('Plot: Features dataset')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```




```
#splitting dataset into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

```
#Inherting Callback class from tensorflow
class get_Metrics(tf.keras.callbacks.Callback):
  def __init__(self):
    self.validation_data=(X_test,y_test)
  def on_train_begin(self, logs={}):
    self.f1_value_list = []
  def on_epoch_end(self, epoch, logs={}):
     predict_value = (np.asarray(self.model.predict(self.validation_data[0]))).round()
     target_value = self.validation_data[1]
     f1_value = f1_score(target_value, predict_value.round())
```

```
     roc_val=roc_auc_score(target_value, predict_value)
     self.f1_value_list.append(f1_value)
     print("--> f1 score :{}  --> ROCValue : {}".format(f1_value, roc_val))
```

```
class get_Terminate_NaN(tf.keras.callbacks.Callback):
    """If you are getting any NaN values(either weigths or loss) while training, you have to terminate your training."""
    def on_epoch_end(self, epoch, logs={}):
        loss_value = logs.get('loss')
        if loss_value is not None:
            if np.isinf(loss_value) or np.isnan(loss_value):
                print("Invalid loss and terminated at epoch",epoch)
                self.model.stop_training = True
```

```
def get_lr_scheduler(epoch, lr):
    step_decay = 3
    rate_of_decay = 0.95

    if (epoch+1) % step_decay == 0 :
        res = lr * rate_of_decay
        return res
    return lr
```

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.

```
# Load the TensorBoard notebook extension
%load_ext tensorboard
```

```
!rm -rf ./logs/
```

```
def model_1():

    input_layer = tf.keras.layers.Dense(2,activation="tanh",input_shape=(2,),kernel_initializer=keras.initializers.RandomUniform(minval=-
    dense_layer_1 = tf.keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1))
    dense_layer_2 = tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1)
    dense_layer_3 = tf.keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1))
    dense_layer_4 = tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1)
    dense_layer_5 = tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1)
    output_layer  = tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1
    model_args = tf.keras.models.Sequential([input_layer,dense_layer_1, dense_layer_2,dense_layer_3, dense_layer_4, dense_layer_5, output

    return model_args
```

```
#Definig each parameters of the model
file_path="model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
l_r_schedule = LearningRateScheduler(get_lr_scheduler, verbose=0)
reduce_l_r = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9, patience=1, min_lr=0.0001)
checkpoint = ModelCheckpoint(filepath=file_path, monitor='val_accuracy',  verbose=1, save_best_only=True, mode='auto')
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.35, patience=2, verbose=1)
metrics=get_Metrics()
terminate= get_Terminate_NaN()
log_dir_files = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir_files, histogram_freq=1)
```

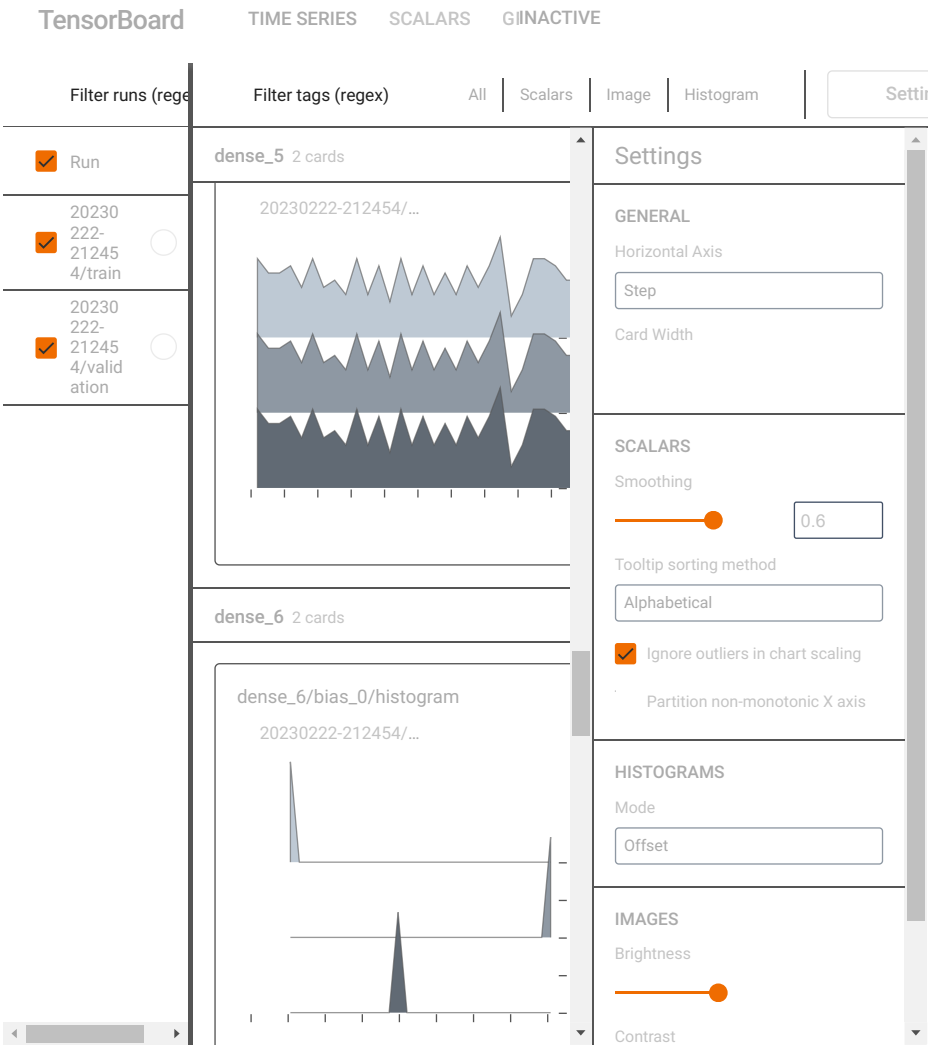
```
m_1=model_1()
optimizer_m_1=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, name='SGD')
m_1.compile(optimizer_m_1,loss='BinaryCrossentropy', metrics=['accuracy'])
m_1.fit(x=X_train, y=y_train, epochs=30,
          validation_data=(X_test, y_test),callbacks=[metrics,checkpoint,terminate,l_r_schedule ,reduce_l_r,tensorboard_callback,earlystc
```

```
     Epoch 1/30
       6/438 [..............................] - ETA: 4s - loss: 3.7322 - accuracy: 0.5260   WARNING:tensorflow:Callback method `on_train
     188/188 [==============================] - 1s 5ms/step
```

```
--> f1 score :0.6728599867285998  --> ROCValue : 0.5

Epoch 1: val_accuracy improved from -inf to 0.50700, saving model to model_save/weights-01-0.5070.hdf5
438/438 [==============================] - 10s 13ms/step - loss: 1.0860 - accuracy: 0.4970 - val_loss: 0.6935 - val_accuracy: 0.507
Epoch 2/30
188/188 [==============================] - 2s 9ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5

Epoch 2: val_accuracy did not improve from 0.50700
438/438 [==============================] - 7s 15ms/step - loss: 0.6933 - accuracy: 0.4970 - val_loss: 0.6935 - val_accuracy: 0.5070
Epoch 3/30
188/188 [==============================] - 2s 8ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5

Epoch 3: val_accuracy did not improve from 0.50700
438/438 [==============================] - 8s 19ms/step - loss: 0.6936 - accuracy: 0.4970 - val_loss: 0.6931 - val_accuracy: 0.5070
Epoch 3: early stopping
<keras.callbacks.History at 0x7facd8400370>
```

```
# %reload_ext tensorboard
```

```
%tensorboard --logdir logs
```




```
!rm -rf ./logs/
```

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.

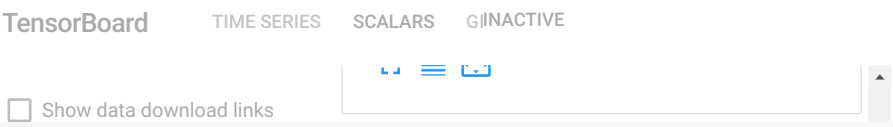
```
def model_2():

    input_layer = tf.keras.layers.Dense(2,activation="relu",input_shape=(2,),kernel_initializer=keras.initializers.RandomUniform(minval=-
    dense_layer_1 = tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1))
    dense_layer_2 = tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1)
    dense_layer_3 = tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1))
    dense_layer_4 = tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1)
    dense_layer_5 = tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1)
    output_layer  = tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers.RandomUniform(minval=-0, maxval=1
    model_2_args = tf.keras.models.Sequential([input_layer,dense_layer_1, dense_layer_2,dense_layer_3, dense_layer_4, dense_layer_5, outp

    return model_2_args
```

```
m_2=model_2()
optimizer_m_2=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, name='SGD')
m_2.compile(optimizer_m_2, loss='BinaryCrossentropy',  metrics=['accuracy'])
m_2.fit(x=X_train, y=y_train, epochs=30,
          validation_data=(X_test, y_test),callbacks=[checkpoint,earlystop,terminate,l_r_schedule ,reduce_l_r,metrics,tensorboard_callbac
```

```
Epoch 1/30
  1/438 [..............................] - ETA: 6:58 - loss: 2749.5400 - accuracy: 0.4375WARNING:tensorflow:Callback method `on_tra
425/438 [============================>.] - ETA: 0s - loss: 7.1610 - accuracy: 0.4962
Epoch 1: val_accuracy did not improve from 0.50700
188/188 [==============================] - 0s 2ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 4s 7ms/step - loss: 6.9762 - accuracy: 0.4970 - val_loss: 0.6932 - val_accuracy: 0.5070
Epoch 2/30
437/438 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy: 0.4971
Epoch 2: val_accuracy did not improve from 0.50700
188/188 [==============================] - 1s 3ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 3s 6ms/step - loss: 0.6932 - accuracy: 0.4970 - val_loss: 0.6932 - val_accuracy: 0.5070
Epoch 3/30
435/438 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy: 0.4971
Epoch 3: val_accuracy did not improve from 0.50700
188/188 [==============================] - 1s 3ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 4s 10ms/step - loss: 0.6932 - accuracy: 0.4970 - val_loss: 0.6932 - val_accuracy: 0.5070
Epoch 3: early stopping
<keras.callbacks.History at 0x7facd8a48280>
```

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 1539), started 0:13:10 ago. (Use '!kill 1539' to
kill it.)
```




```
!rm -rf ./logs/
```

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.

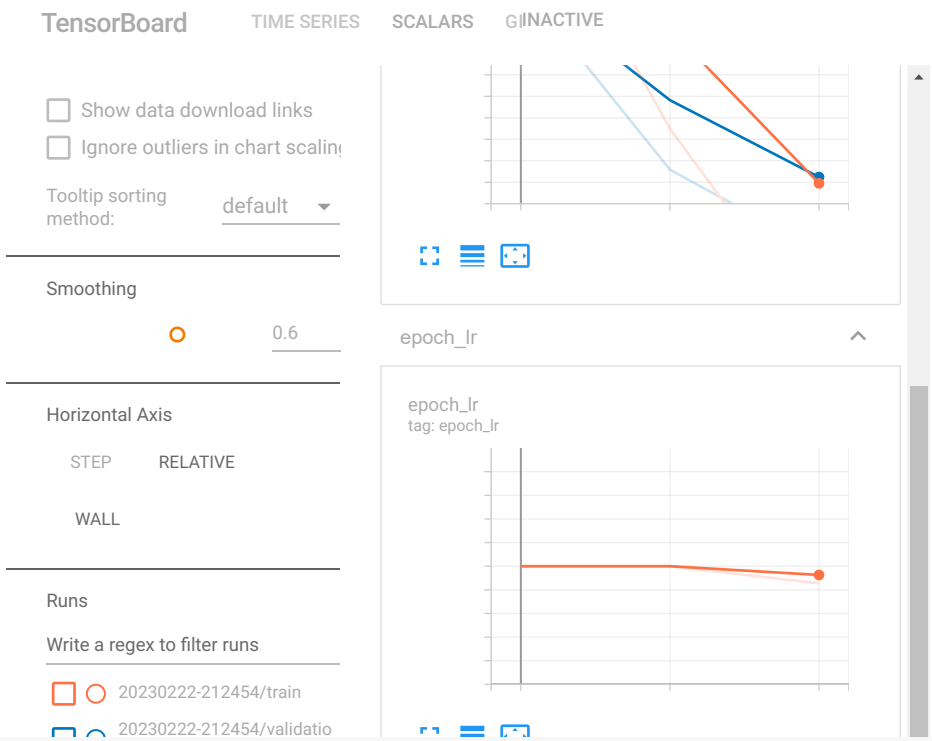
```
def model_3():

    input_layer = tf.keras.layers.Dense(2,activation="relu",input_shape=(2,),kernel_initializer=keras.initializers.he_uniform())
    dense_layer_1 = tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_2 = tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_3 = tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_4 = tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_5 = tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uniform())
    output_layer  = tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers.he_uniform())
    model_3_args = tf.keras.models.Sequential([input_layer,dense_layer_1, dense_layer_2,dense_layer_3, dense_layer_4, dense_layer_5, outp

    return model_3_args
```

```
m_3=model_3()
optimizer_m_3=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, name='SGD')
m_3.compile(optimizer_m_3, loss='BinaryCrossentropy', metrics=['accuracy'])
m_3.fit(x=X_train, y=y_train, epochs=30, validation_data=(X_test, y_test),callbacks=[checkpoint,earlystop,terminate,l_r_schedule ,reduce_
```

```
Epoch 1/30
  1/438 [..............................] - ETA: 7:56 - loss: 0.6775 - accuracy: 0.5000WARNING:tensorflow:Callback method `on_train_
437/438 [============================>.] - ETA: 0s - loss: 0.6880 - accuracy: 0.4970
Epoch 1: val_accuracy did not improve from 0.50700
188/188 [==============================] - 1s 3ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 6s 11ms/step - loss: 0.6879 - accuracy: 0.4970 - val_loss: 0.6851 - val_accuracy: 0.5070
Epoch 2/30
435/438 [============================>.] - ETA: 0s - loss: 0.6818 - accuracy: 0.4970
Epoch 2: val_accuracy did not improve from 0.50700
188/188 [==============================] - 0s 2ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 3s 6ms/step - loss: 0.6817 - accuracy: 0.4970 - val_loss: 0.6808 - val_accuracy: 0.5070
Epoch 3/30
434/438 [============================>.] - ETA: 0s - loss: 0.6769 - accuracy: 0.4971
Epoch 3: val_accuracy did not improve from 0.50700
188/188 [==============================] - 0s 2ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 3s 6ms/step - loss: 0.6770 - accuracy: 0.4970 - val_loss: 0.6789 - val_accuracy: 0.5070
Epoch 3: early stopping
<keras.callbacks.History at 0x7facd8896460>
```

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 1539), started 0:17:41 ago. (Use '!kill 1539' to
kill it.)
```




```
!rm -rf ./logs/
```

Model-4

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

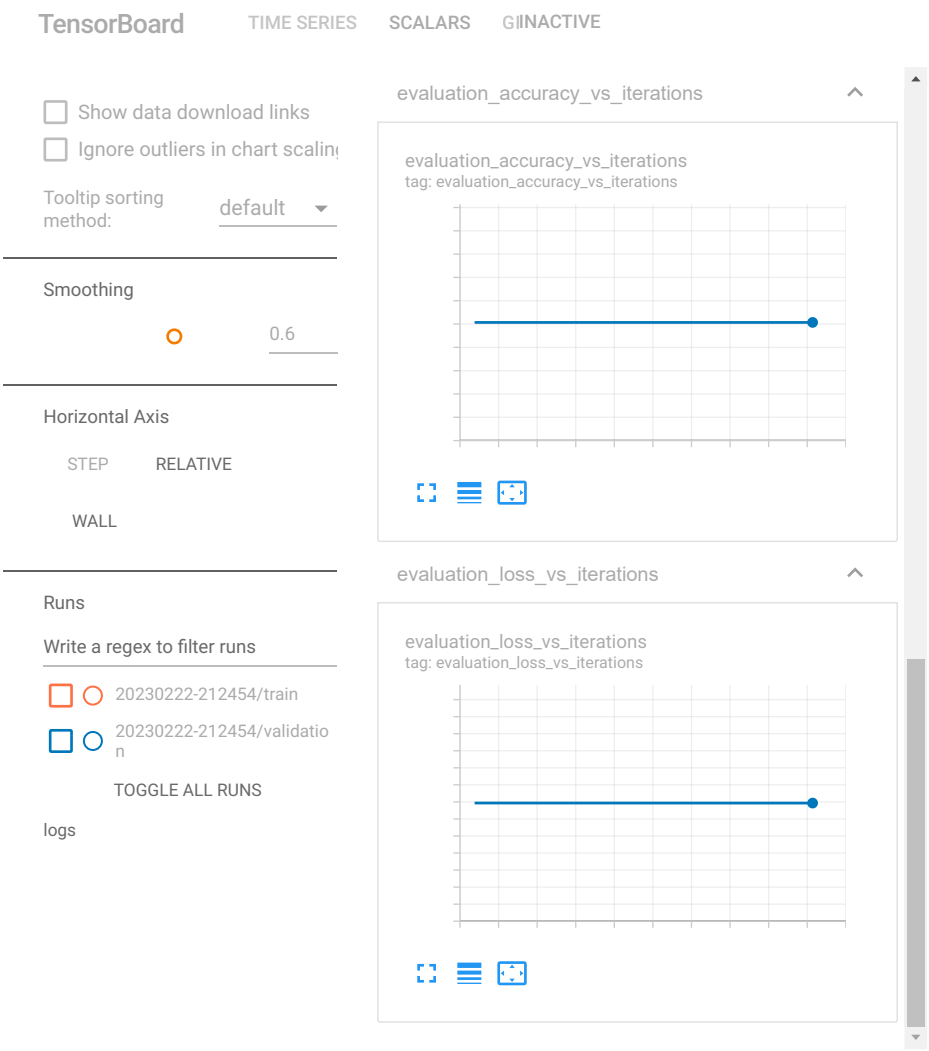
```
def model_4():

    input_layer = tf.keras.layers.Dense(2,activation="sigmoid",input_shape=(2,),kernel_initializer=keras.initializers.he_uniform())
    dense_layer_1 = tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_2 = tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_3 = tf.keras.layers.Dense(16, activation="sigmoid",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_4 = tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uniform())
    dense_layer_5 = tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.he_uniform())
    output_layer  = tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers.he_uniform())
    model_4_args = tf.keras.models.Sequential([input_layer,dense_layer_1, dense_layer_2,dense_layer_3, dense_layer_4, dense_layer_5, outp

    return model_4_args
```

```
m_4=model_4()
optimizer_m4=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, name='SGD')
m_4.compile(optimizer_m4, loss='BinaryCrossentropy', metrics=['accuracy'])
m_4.fit(x=X_train, y=y_train, epochs=30,
          validation_data=(X_test, y_test),callbacks=[checkpoint,earlystop,terminate,l_r_schedule ,reduce_l_r,metrics,tensorboard_callbac
```

```
Epoch 1/30
  1/438 [..............................] - ETA: 11:21 - loss: 1.4165 - accuracy: 0.4375WARNING:tensorflow:Callback method `on_train
427/438 [============================>.] - ETA: 0s - loss: 0.7078 - accuracy: 0.4967
Epoch 1: val_accuracy did not improve from 0.50700
188/188 [==============================] - 1s 2ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 5s 9ms/step - loss: 0.7075 - accuracy: 0.4970 - val_loss: 0.6933 - val_accuracy: 0.5070
Epoch 2/30
431/438 [============================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.4967
Epoch 2: val_accuracy did not improve from 0.50700
188/188 [==============================] - 0s 2ms/step
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 3s 6ms/step - loss: 0.6937 - accuracy: 0.4970 - val_loss: 0.6931 - val_accuracy: 0.5070
Epoch 3/30
438/438 [==============================] - ETA: 0s - loss: 0.6934 - accuracy: 0.4970
Epoch 3: val_accuracy did not improve from 0.50700
188/188 [==============================] - 0s 2ms/step
```

```
--> f1 score :0.6728599867285998  --> ROCValue : 0.5
438/438 [==============================] - 2s 5ms/step - loss: 0.6934 - accuracy: 0.4970 - val_loss: 0.6928 - val_accuracy: 0.5070
Epoch 3: early stopping
<keras.callbacks.History at 0x7facd845bdc0>
```

```
%tensorboard --logdir logs
```




```
!rm -rf ./logs/
```

✓ 6s completed at 10:48 PM