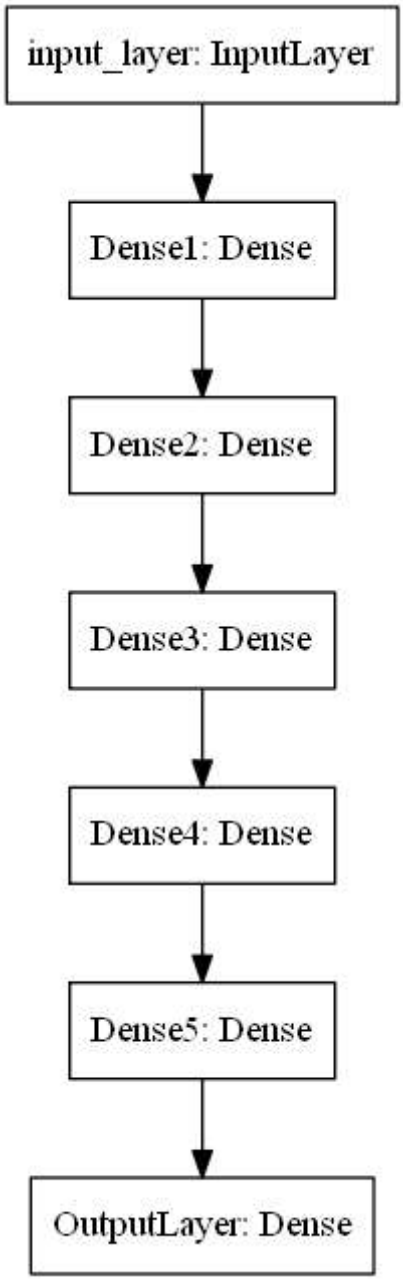
1. Download the data from here. You have to use data.csv file for this assignment

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 accu
       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.

importing the libraries

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
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
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import LearningRateScheduler

!wget --header="Host: doc-08-3k-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0
```

```
--2022-11-14 03:40:15--  https://doc-08-3k-docs.googleusercontent.com/docs/securesc/8h8
Resolving doc-08-3k-docs.googleusercontent.com (doc-08-3k-docs.googleusercontent.com)..
Connecting to doc-08-3k-docs.googleusercontent.com (doc-08-3k-docs.googleusercontent.cor
HTTP request sent, awaiting response... 403 Forbidden
2022-11-14 03:40:15 ERROR 403: Forbidden.
```

importing the dataset

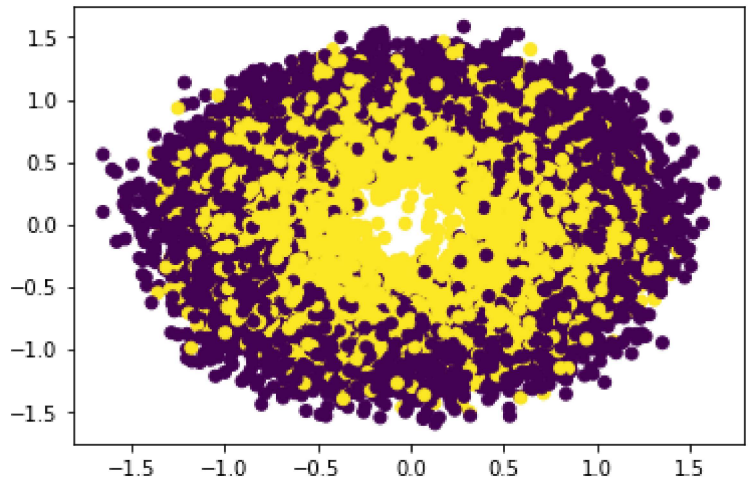
```
data=pd.read_csv("data.csv")
data.head()
```

	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

```
y= data ['label'].values
x= data [['f1','f2']].values
```

plotted scatter plot to show the dataset distribution

```
import matplotlib.pyplot as plt

# plotting the scatter plot
plt.scatter(data['f1'], data['f2'], c=y)
plt.show()
```



splitting the dataset

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

IN below code i intilizing the callbacks code

Ref : https://deepnotes.io/sgd-momentum-adaptive

```
class Metrics(tf.keras.callbacks.Callback):
  def __init__(self):
    self.validation_data=(x_test,y_test)
  def on_train_begin(self, logs={}):
    self.value_f1_score = []
  def on_epoch_end(self, epoch, logs={}):

# computing the predict value from dataset
     value_predict = (self.model.predict(self.validation_data[0]))

     # computing the targeted value from validation data
     value_target = self.validation_data[1]

     # computing the f1 score with micro f1 score
     value_f1 = f1_score(value_target, value_predict.round() ,average ='micro')

     # computing the roc value from auc score
     roc_value=roc_auc_score(value_target, value_predict)

    # now will append the value in f1 score
```

```
        self.value_f1_score.append(value_f1)
        print("-f1 score :",value_f1,"-ROCValue :", roc_value)
```

now this is time for set the condition for call_backs

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

    def epoch_end(self, epoch, logs={},model={}):
        loss = logs.get('loss')
        model_weights = self.model.get_weights()
        # here we are setting the conditon that if our loss will nan or infinity than stop le
        if loss is not None:
            if np.isnan(loss) or np.isinf(loss):
                print("if got not_valid loss than teraminate at  epoch {}".format(epoch))
          # setting the condition if weight value will be nan then stop traning
        if model_weights is not None:
            if np.any([np.any(np.isnan(x)) for x in model_weights]):
                print("if got not_valid weight than teraminate at  epoch {}".format(epoch))

                # stop the model traning
                self.model.stop_training = True


def learning_rate_scheduler(epoch, learning_rate):
    decay_rate = .97
    decay_step = 3

    # setting the condtion for learning rate
    if (epoch+1) % decay_step == 0 :
        return learning_rate * decay_rate
    return learning_rate


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

as per instructin using the tanh function for all layer except output layer

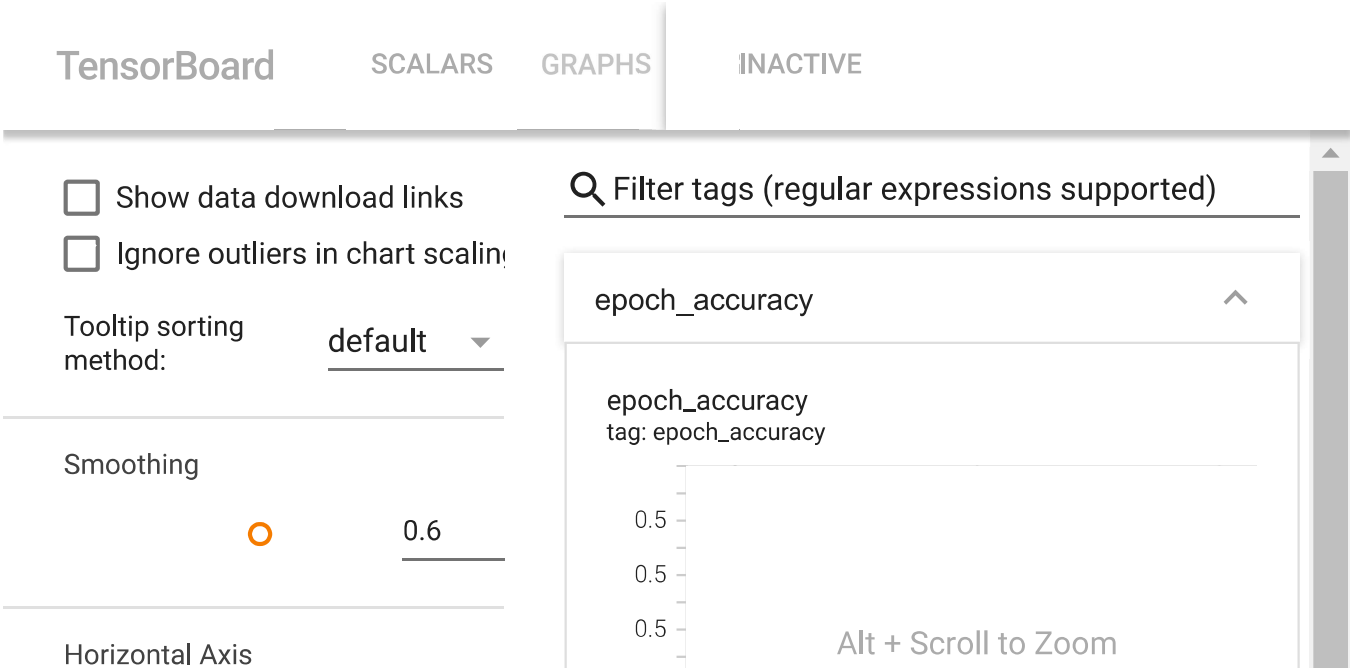
using randomuniform intilizer as per instruction

```
def create_model_1():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(2,activation="tanh",input_shape=(2,),kernel_initializer=keras.initi
    tf.keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.RandomU
    tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.Random
    tf.keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.RandomU
    tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.Random
    tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.Random
    tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers.Rando
  ])


filepath="model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"

learning_rate_schedule = LearningRateScheduler(learning_rate_scheduler, verbose=0)

# here we are saving the model weight if the accuracy improves
#REF : https://machinelearningmastery.com/check-point-deep-learning-models-keras/
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_best

#  HERE WITH HELP OF REDUCE_LR WE ARE STOPPING OUR LEARNING RATE AFTER NOT IMPROVEMENT IN ACC
#REF: https://stackoverflow.com/questions/51889378/how-to-use-keras-reducelronplateau
reduce_learning_rate = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9, patience=1, min_

# using early stopping if the model got the certain condtion
# REF : https://stackoverflow.com/questions/50284898/keras-earlystopping-which-min-delta-and-
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.35, patience=2, verbose=1)

terminate= TerminateNaN()
metrics=Metrics()


logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)


model_1=create_model_1()

# as per instruction using sgd optimizer
# starting the learning rate from 0.01

optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name='SGD

# using binary cross entropy as loss function for this model
model_1.compile(optimizer,loss='BinaryCrossentropy', metrics=['accuracy'])

# fitting the data x_train and y_train
# most imp thing you can see than after all the model paramters we are setting_up our all the
model_1.fit(x=x_train,
          y=y_train,
          epochs=15,
```

```
        validation_data=(x_test, y_test),callbacks=[metrics,checkpoint,terminate,learning_r
        )
```

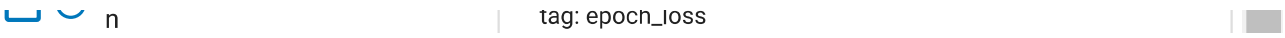
```
Epoch 1/15
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.4996 -ROCValue : 0.5

Epoch 1: val_accuracy improved from -inf to 0.49960, saving model to model_save/weights.
469/469 [==============================] - 3s 5ms/step - loss: 0.7667 - accuracy: 0.5001
Epoch 2/15
157/157 [==============================] - 0s 2ms/step
-f1 score : 0.4996 -ROCValue : 0.5

Epoch 2: val_accuracy did not improve from 0.49960
469/469 [==============================] - 2s 3ms/step - loss: 0.6929 - accuracy: 0.5001
Epoch 3/15
157/157 [==============================] - 0s 2ms/step
-f1 score : 0.4996 -ROCValue : 0.5

Epoch 3: val_accuracy did not improve from 0.49960
469/469 [==============================] - 2s 4ms/step - loss: 0.6938 - accuracy: 0.5001
Epoch 3: early stopping
<keras.callbacks.History at 0x7f4bc1930cd0>
```

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



You can see that there is not big diff between train validation accuracy so we not overfitting

after 1st epoch we have got accuracy of 50.01 and there is no improvement



No big improvement in loss and f1 score after 2nd epoch so will be stop here

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.

As per instruction using relu activation function except output layer

using randomuniorm intilizer

we are using model like model 1 except relu activation function

```
def create_model_2():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(2,activation="relu",input_shape=(2,),kernel_initializer=keras.initi
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.RandomN
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.Random
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.RandomN
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.Random
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.Random
    tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers.Rando
  ])
```

using binary cross entropy as loss function

using sgd optimizer

matrics using accuracy

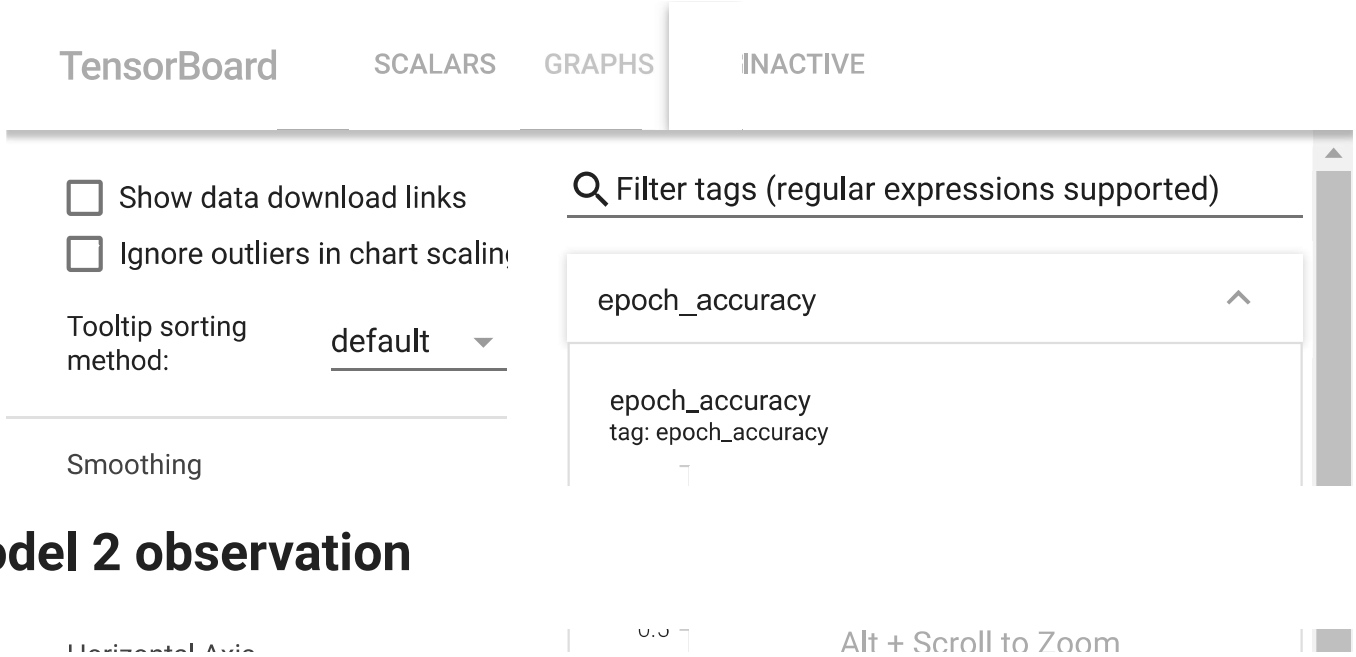
```
model_2=create_model_2()
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name='SGD
model_2.compile(optimizer,
              loss='BinaryCrossentropy',        # loss function as binary cross entropy
              metrics=['accuracy'])  # using accuracy as matric

# using relu as activation function
model_2.fit(x=x_train,
          y=y_train,
          epochs=10,
          validation_data=(x_test, y_test),callbacks=[checkpoint,earlystop,terminate,learning
          )
```

```
Epoch 1/10
  1/469 [..............................] - ETA: 4:52 - loss: 127.4772 - accuracy: 0.3750
```

```
451/469 [===========================>..] - ETA: 0s - loss: 1.1713 - accuracy: 0.5001
Epoch 1: val_accuracy did not improve from 0.49960
157/157 [==============================] - 0s 2ms/step
-f1 score : 0.4996 -ROCValue : 0.5
469/469 [==============================] - 3s 4ms/step - loss: 1.1533 - accuracy: 0.500
Epoch 2/10
460/469 [===========================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.5005
Epoch 2: val_accuracy did not improve from 0.49960
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.4996 -ROCValue : 0.5
469/469 [==============================] - 2s 4ms/step - loss: 0.6937 - accuracy: 0.500
Epoch 3/10
459/469 [===========================>.] - ETA: 0s - loss: 0.6932 - accuracy: 0.5011
Epoch 3: val_accuracy did not improve from 0.49960
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.4996 -ROCValue : 0.5
469/469 [==============================] - 2s 4ms/step - loss: 0.6932 - accuracy: 0.500
Epoch 3: early stopping
<keras.callbacks.History at 0x7f4bbe011e50>
```

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

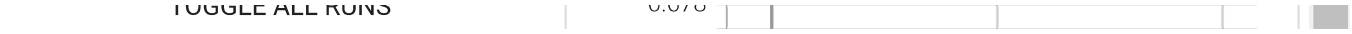
```
Reusing TensorBoard on port 6006 (pid 186), started 0:00:16 ago. (Use '!kill 186' to
kill it.)
```



Model 2 observation

as we can see ater 2nd epoch there is not much improvement in loss function and accuracy

In model 2 we have got 50.01% accuracy with loss of 69.32% after that model stop using earlystoping with the help of call_backs



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.

here also we have used relu activation function except output layer as per instrunction given for model3

In model 3 we are using he intilizer

here also we are using a sgd as optimizer

```
def create_model_3():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(2,activation="relu",input_shape=(2,),kernel_initializer=keras.initi
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_unif
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uni
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_unif
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uni
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.he_uni
    tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers.he_un
  ])


model_3=create_model_3()
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name='SGD

# here also using binary cross entorphy
# matric we are using is accuracy here
model_3.compile(optimizer,
              loss='BinaryCrossentropy',
              metrics=['accuracy'])

# here we are using he intilizers with sgd optimizer
model_3.fit(x=x_train,
          y=y_train,
          epochs=5,
          validation_data=(x_test, y_test),callbacks=[checkpoint,earlystop,terminate,learning
          )
```

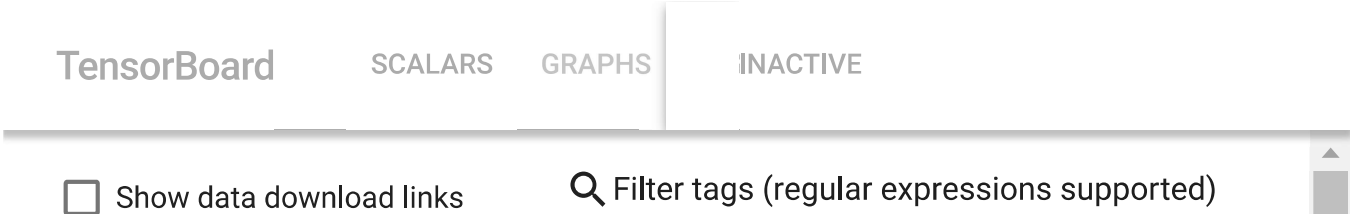
```
Epoch 1/5
  1/469 [..............................] - ETA: 4:36 - loss: 0.6912 - accuracy: 0.4688W
459/469 [============================>.] - ETA: 0s - loss: 0.6900 - accuracy: 0.5005
Epoch 1: val_accuracy did not improve from 0.49960
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.4996 -ROCValue : 0.5
469/469 [==============================] - 2s 4ms/step - loss: 0.6900 - accuracy: 0.500
Epoch 2/5
445/469 [===========================>..] - ETA: 0s - loss: 0.6773 - accuracy: 0.4989
Epoch 2: val_accuracy did not improve from 0.49960
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.4996 -ROCValue : 0.5
469/469 [==============================] - 2s 3ms/step - loss: 0.6772 - accuracy: 0.500
```

```
Epoch 3/5
445/469 [===========================>..] - ETA: 0s - loss: 0.6699 - accuracy: 0.5007
Epoch 3: val_accuracy did not improve from 0.49960
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.4996 -ROCValue : 0.5
469/469 [==============================] - 2s 4ms/step - loss: 0.6695 - accuracy: 0.5001
Epoch 3: early stopping
<keras.callbacks.History at 0x7f4bc164e850>
```

Double-click (or enter) to edit

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

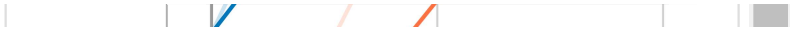
```
Reusing TensorBoard on port 6006 (pid 186), started 0:00:22 ago. (Use '!kill 186' to
kill it.)
```



here we got end-up with the accuracy 50.9% with loss o 66.95%.

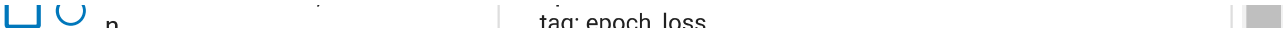
not much improvement almost negligible

no big change or improvement in model3 except littile bit in loss



Model-4

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



As we for model-4 we have decrease the number of neurons in layers as per instruction

using adam optimizer

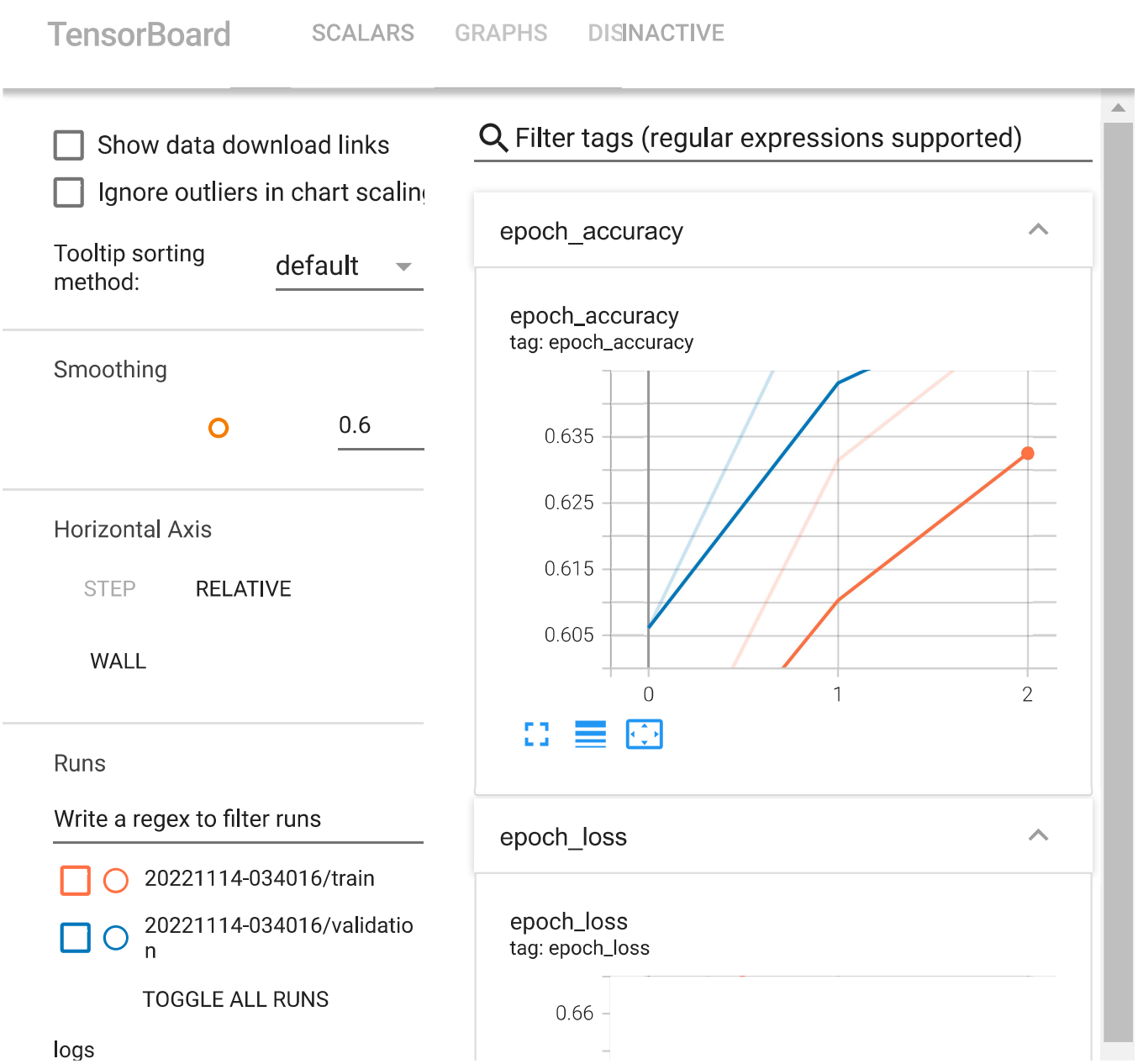

```
def create_model_4():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(4,activation="relu",input_shape=(2,),kernel_initializer=keras.initi

    # here you can see that we have decrease the value of number of neurons connected to laye
    tf.keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.he_unifo
    tf. keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.he_unif
```

```
    tf.keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.he_unifo
    tf. keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.he_unif
    tf. keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.he_unif
    tf.keras.layers.Dense(1, activation='sigmoid',kernel_initializer=keras.initializers.he_un
  ])

model_4=create_model_4()

# using adam optimizer
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False, name='ada
model_4.compile(optimizer,
              loss='BinaryCrossentropy',
              metrics=['accuracy'])
model_4.fit(x=x_train,
          y=y_train,
          epochs=5,
          validation_data=(x_test, y_test),callbacks=[checkpoint,earlystop,terminate,learning
          )
```

```
Epoch 1/5
  1/469 [..............................] - ETA: 4:35 - loss: 0.6978 - accuracy: 0.4062W
456/469 [============================>.] - ETA: 0s - loss: 0.6750 - accuracy: 0.5745
Epoch 1: val_accuracy improved from 0.49960 to 0.60620, saving model to model_save/weigh
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.6062 -ROCValue : 0.6677407473540783
469/469 [==============================] - 3s 4ms/step - loss: 0.6751 - accuracy: 0.575
Epoch 2/5
447/469 [===========================>..] - ETA: 0s - loss: 0.6440 - accuracy: 0.6292
Epoch 2: val_accuracy improved from 0.60620 to 0.66520, saving model to model_save/weigh
157/157 [==============================] - 0s 2ms/step
-f1 score : 0.6652 -ROCValue : 0.7389406329220051
469/469 [==============================] - 2s 4ms/step - loss: 0.6422 - accuracy: 0.631
Epoch 3/5
444/469 [===========================>..] - ETA: 0s - loss: 0.6209 - accuracy: 0.6536
Epoch 3: val_accuracy improved from 0.66520 to 0.66920, saving model to model_save/weigh
157/157 [==============================] - 0s 1ms/step
-f1 score : 0.6692 -ROCValue : 0.7481942388443129
469/469 [==============================] - 2s 4ms/step - loss: 0.6205 - accuracy: 0.653
Epoch 3: early stopping
<keras.callbacks.History at 0x7f4bc1420a90>
```

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

```
Reusing TensorBoard on port 6006 (pid 186), started 0:00:28 ago. (Use '!kill 186' to
kill it.)
```



yes there is significant improvement in loss 3% improvement in loss

accuracy we have got 65% huge improvement from 50% to 65%

CONCLUSION

1. changing the activation function not improving in accurracy and loss

2. there is huge improvement when we have decrease the number of values

3. 3% imporvment in loss and 15% improvement in accuracy

4. rather changing the activation unction we should change the value or we can regulate the value to get the better value