```
# all needed packages
import pandas as pd
import math as math
import numpy as np
import json, csv, itertools
from matplotlib import pyplot

# Machine learning packages
from sklearn import linear_model, ensemble, metrics
from sklearn.model_selection import cross_val_score,KFold,StratifiedKFold
from sklearn.metrics import *
from xgboost import XGBClassifier
import xgboost as xgb
from sklearn.ensemble import  RandomForestClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import MinMaxScaler, MaxAbsScaler

# DL Keras
from tensorflow import keras
from tensorflow.keras import layers
from keras.regularizers import l2
from keras.layers import Input, Reshape, Dense, Dropout, concatenate
from keras.models import Sequential, Model

import tensorflow as tf

from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import roc_curve, roc_auc_score
from itertools import chain

import matplotlib.pyplot as pyplot


#Define 10 cancer types
cancertypes = ["Bladder","Breast", "Colon", "Kidney", "Liver","Luk",
               "Lung", "NonHodgkinLymph", "Rectal", "Thyroid" ]


## Define the DL Model
DL_model = Sequential()
DL_model.add(Dense(64, input_dim=5, kernel_initializer='normal', activation='tanh', kernel_regularizer=l2(0.01), bias_regu
# DL_model.add(Dropout(0.2))
DL_model.add(Dense(32,  activation='tanh', kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
DL_model.add(Dense(1, activation='sigmoid'))

## Define 10-fold CV
skf = StratifiedKFold(n_splits=10, shuffle = True, random_state = 10)


DL_omCancerAUC = []
DL_omCancerFPR= []
DL_omCancerTPR= []

for cancer in cancertypes:
    print('\n*Working with '+ cancer + " Cancer*" )

    ## Read  combined genes (positive and negative genes) per Cancer
    #Genes = pd.read_csv('Datasets/combinedGenes_cancer/'+str(cancer)+'_Genes.csv')
    Bladder=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Bladder_Genes.csv")
    Breast=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Breast_Genes.csv")
    Colon=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Colon_Genes.csv")
    Kidney=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Kidney_Genes.csv")
    Liver=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Liver_Genes.csv")
    Luk=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Luk_Genes.csv")
    Lung=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Lung_Genes.csv")
    NonHodgkinLymph=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/NonHodgkinLymph_Genes.csv")
    Rectal=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Rectal_Genes.csv")
    Thyroid=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Thyroid_Genes.csv")
```

```
    ## Read generated BERT embeddings for each genes
    #Embed = np.genfromtxt('BERT_Embed/'+str(cancer)+'_Embed.txt')
    Bladder_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Bladder_Embed.txt", usecols=range(679
    Breast_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Breast_Embed.txt", usecols=range(422))
    Colon_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Colon_Embed.txt", usecols=range(411))
    Kidney_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Kidney_Embed.txt", usecols=range(413))
    Liver_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Liver_Embed.txt", usecols=range(694))
    Luk_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Luk_Embed.txt", usecols=range(270))
    Lung_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Lung_Embed.txt", usecols=range(545))
    NonHodgkinLymph_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/NonHodgkinLymph_Embed.txt", u
    Rectal_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Rectal_Embed.txt", usecols=range(686))
    Thyroid_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Thyroid_Embed.txt", usecols=range(556

    #Embed_df = pd.DataFrame(Embed)
    Bladder_Embed_df = pd.DataFrame(Bladder_Embed)
    Breast_Embed_df = pd.DataFrame(Breast_Embed)
    Colon_Embed_df = pd.DataFrame(Colon_Embed)
    Kidney_Embed_df = pd.DataFrame(Kidney_Embed)
    Liver_Embed_df = pd.DataFrame(Liver_Embed)
    Luk_Embed_df = pd.DataFrame(Luk_Embed)
    Lung_Embed_df = pd.DataFrame(Lung_Embed)
    NonHodgkinLymph_Embed_df = pd.DataFrame(NonHodgkinLymph_Embed)
    Rectal_Embed_df = pd.DataFrame(Rectal_Embed)
    Thyroid_Embed_df = pd.DataFrame(Thyroid_Embed)

    ## The embeddings have the same order of the combined genes,
    #  so concat them using the index
    #allG_em = pd.merge(Genes, Embed_df,right_index=True,left_index=True)
    # ...


    Bladder_allG_em = pd.concat([Bladder, Bladder_Embed_df], axis=1)
    Breast_allG_em = pd.concat([Breast, Breast_Embed_df], axis=1)
    Colon_allG_em = pd.concat([Colon, Colon_Embed_df], axis=1)
    Kidney_allG_em = pd.concat([Kidney, Kidney_Embed_df], axis=1)
    Liver_allG_em = pd.concat([Liver, Liver_Embed_df], axis=1)
    Luk_allG_em = pd.concat([Luk, Luk_Embed_df], axis=1)
    Lung_allG_em = pd.concat([Lung, Lung_Embed_df], axis=1)
    NonHodgkinLymph_allG_em = pd.concat([NonHodgkinLymph, NonHodgkinLymph_Embed_df], axis=1)
    Rectal_allG_em = pd.concat([Rectal, Rectal_Embed_df], axis=1)
    Thyroid_allG_em = pd.concat([Thyroid, Thyroid_Embed_df], axis=1)

# ...

    ## Read gene mutation and gene expression
    #GE = pd.read_csv('Datasets/OMICS/'+str(cancer)+'_gene_exp.csv')
    Bladder_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Bladder_gene_exp.csv")
    Breast_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Breast_gene_exp.csv")
    Colon_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Colon_gene_exp.csv")
    Kidney_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Kidney_gene_exp.csv")
    Liver_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Liver_gene_exp.csv")
    Luk_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Luk_gene_exp.csv")
    Lung_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Lung_gene_exp.csv")
    NonHodgkinLymph_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/NonHodgkinLymph_gene_exp.csv")
    Rectal_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Rectal_gene_exp.csv")
    Thyroid_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Thyroid_gene_exp.csv")

    #Mu = pd.read_csv('Datasets/OMICS/'+str(cancer)+'_gene_mut.csv')
    Bladder_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Bladder_gene_mut.csv")
    Breast_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Breast_gene_mut.csv")
    Colon_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Colon_gene_mut.csv")
    Kidney_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Kidney_gene_mut.csv")
    Liver_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Liver_gene_mut.csv")
    Luk_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Luk_gene_mut.csv")
    Lung_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Lung_gene_mut.csv")
    NonHodgkinLymph_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/NonHodgkinLymph_gene_mut.csv")
    Rectal_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Rectal_gene_mut.csv")
    Thyroid_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Thyroid_gene_mut.csv")
```

```
# merge OMICS features and remove the samples GE values

#OMICS = pd.merge(GE,MU, on=['Gene'], how='inner')
Bladder_gene_OMICS = pd.concat([Bladder_gene_exp, Bladder_gene_mut], axis=1)
Breast_gene_OMICS = pd.concat([Breast_gene_exp, Breast_gene_mut], axis=1)
Colon_gene_OMICS = pd.concat([Colon_gene_exp, Colon_gene_mut], axis=1)
Kidney_gene_OMICS = pd.concat([Kidney_gene_exp, Kidney_gene_mut], axis=1)
Liver_gene_OMICS = pd.concat([Liver_gene_exp, Liver_gene_mut], axis=1)
Luk_gene_OMICS = pd.concat([Luk_gene_exp, Luk_gene_mut], axis=1)
Lung_gene_OMICS = pd.concat([Lung_gene_exp, Lung_gene_mut], axis=1)
NonHodgkinLymph_gene_OMICS = pd.concat([NonHodgkinLymph_gene_exp, NonHodgkinLymph_gene_mut], axis=1)
Rectal_gene_OMICS = pd.concat([Rectal_gene_exp, Rectal_gene_mut], axis=1)
Thyroid_gene_OMICS = pd.concat([Thyroid_gene_exp, Thyroid_gene_mut], axis=1)

#CncerOM = OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Bladder_CncerOM = Bladder_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Breast_CncerOM = Breast_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Colon_CncerOM = Colon_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Kidney_CncerOM = Kidney_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Liver_CncerOM = Liver_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Luk_CncerOM = Luk_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Lung_CncerOM = Lung_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
NonHodgkinLymph_CncerOM = NonHodgkinLymph_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Rectal_CncerOM = Rectal_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Thyroid_CncerOM = Thyroid_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]



# integrate all festures and get the lable, remove all genes with no label
#CancerFV = pd.merge(allG_em, CncerOM, on=['Gene'], how='outer')
Bladder_CancerFV = pd.concat([Bladder_allG_em, Bladder_CncerOM],axis=1)
Breast_CancerFV = pd.concat([Breast_allG_em, Breast_CncerOM],axis=1)
Colon_CancerFV = pd.concat([Colon_allG_em, Colon_CncerOM], axis=1)
Kidney_CancerFV = pd.concat([Kidney_allG_em, Kidney_CncerOM], axis=1)
Liver_CancerFV = pd.concat([Liver_allG_em, Liver_CncerOM], axis=1)
Luk_CancerFV = pd.concat([Luk_allG_em, Luk_CncerOM], axis=1)
Lung_CancerFV = pd.concat([Lung_allG_em, Lung_CncerOM], axis=1)
NonHodgkinLymph_CancerFV = pd.concat([NonHodgkinLymph_allG_em, NonHodgkinLymph_CncerOM], axis=1)
Rectal_CancerFV = pd.concat([Rectal_allG_em, Rectal_CncerOM], axis=1)
Thyroid_CancerFV = pd.concat([Thyroid_allG_em, Thyroid_CncerOM], axis=1)

# CancerFV = CancerFV[CancerFV['Label'].notna()]
Bladder_CancerFV = Bladder_CancerFV[Bladder_CancerFV['Label'].notna()]
Breast_CancerFV = Breast_CancerFV[Breast_CancerFV['Label'].notna()]
Colon_CancerFV = Colon_CancerFV[Colon_CancerFV['Label'].notna()]
Kidney_CancerFV = Kidney_CancerFV[Kidney_CancerFV['Label'].notna()]
Liver_CancerFV = Liver_CancerFV[Liver_CancerFV['Label'].notna()]
Luk_CancerFV = Luk_CancerFV[Luk_CancerFV['Label'].notna()]
Lung_CancerFV = Lung_CancerFV[Lung_CancerFV['Label'].notna()]
NonHodgkinLymph_CancerFV = NonHodgkinLymph_CancerFV[NonHodgkinLymph_CancerFV['Label'].notna()]
Rectal_CancerFV = Rectal_CancerFV[Rectal_CancerFV['Label'].notna()]
Thyroid_CancerFV = Thyroid_CancerFV[Thyroid_CancerFV['Label'].notna()]


#CancerFV = CancerFV.fillna(0)
Bladder_CancerFV = Bladder_CancerFV.fillna(0)
Breast_CancerFV = Breast_CancerFV.fillna(0)
Colon_CancerFV = Colon_CancerFV.fillna(0)
Kidney_CancerFV = Kidney_CancerFV.fillna(0)
Liver_CancerFV = Liver_CancerFV.fillna(0)
Luk_CancerFV = Luk_CancerFV.fillna(0)
Lung_CancerFV = Lung_CancerFV.fillna(0)
NonHodgkinLymph_CancerFV = NonHodgkinLymph_CancerFV.fillna(0)
Rectal_CancerFV = Rectal_CancerFV.fillna(0)
Thyroid_CancerFV = Thyroid_CancerFV.fillna(0)

# The FV is only the omics features for omics models
X = np.array(Bladder_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(Breast_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(Colon_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
```

```
X = np.array(Kidney_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(Liver_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(Luk_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(Lung_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(NonHodgkinLymph_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(Rectal_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])
X = np.array(Thyroid_CancerFV[["MAX","AVG","MED","MIN","COUNT"]])


Y = np.array(Bladder_CancerFV['Label'],dtype=int)
Y = np.array(Breast_CancerFV['Label'],dtype=int)
Y = np.array(Colon_CancerFV['Label'],dtype=int)
Y = np.array(Kidney_CancerFV['Label'],dtype=int)
Y = np.array(Liver_CancerFV['Label'],dtype=int)
Y = np.array(Luk_CancerFV['Label'],dtype=int)
Y = np.array(Lung_CancerFV['Label'],dtype=int)
Y = np.array(NonHodgkinLymph_CancerFV['Label'],dtype=int)
Y = np.array(Rectal_CancerFV['Label'],dtype=int)
Y = np.array(Thyroid_CancerFV['Label'],dtype=int)

print('Features and Label shapes', X.shape, Y.shape)
print('++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++')
# all evaluation lists
auc = []
fpr = []
tpr = []

# Compile model
DL_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

foldCounter = 1
# Start training and testing
for train_index, test_index in  skf.split(X,Y):

    print("*** Working with Fold %i :***" %foldCounter)
    #Apply normalization using MaxAbsolute normlization
    min_max_scaler = MinMaxScaler()
    X_train = min_max_scaler.fit(X[train_index])
    X_train_transform = min_max_scaler.transform(X[train_index])
    X_test_transform = min_max_scaler.transform(X[test_index])

    DL_model.fit(X_train_transform, Y[train_index], epochs=20,
             batch_size=16)
    predictedScore = DL_model.predict(X_test_transform)
    #  our model's predictions.
    predictedClass= np.argmax(predictedScore, axis=0)

    #print("@@ Validation and evaluation of fold %i @@" %foldCounter)
    fr, tr, _ = roc_curve(Y[test_index], predictedScore)
    fpr.append(fr)
    tpr.append(tr)
    #print("AUC =  %f" %roc_auc_score(Y[test_index], predictedScore))
    auc.append(roc_auc_score(Y[test_index], predictedScore))

    print('------------------------------------------------------')
    foldCounter += 1

#-------------------------------------------------------
Fpr = np.array(sorted(list(itertools.chain.from_iterable(fpr))))
Tpr = np.array(sorted(list(itertools.chain.from_iterable(tpr))))
#-----------------------------------------------
### Print Evaluation Metrics.....................
print("Results using test data: AUC = " + str( np.array(auc).mean().round(decimals=4) ))
DL_aucROC = np.array(auc).mean().round(decimals=4)
DL_omCancerAUC.append(np.array(DL_aucROC).mean())
DL_omCancerFPR.append(Fpr)
DL_omCancerTPR.append(Tpr)
```

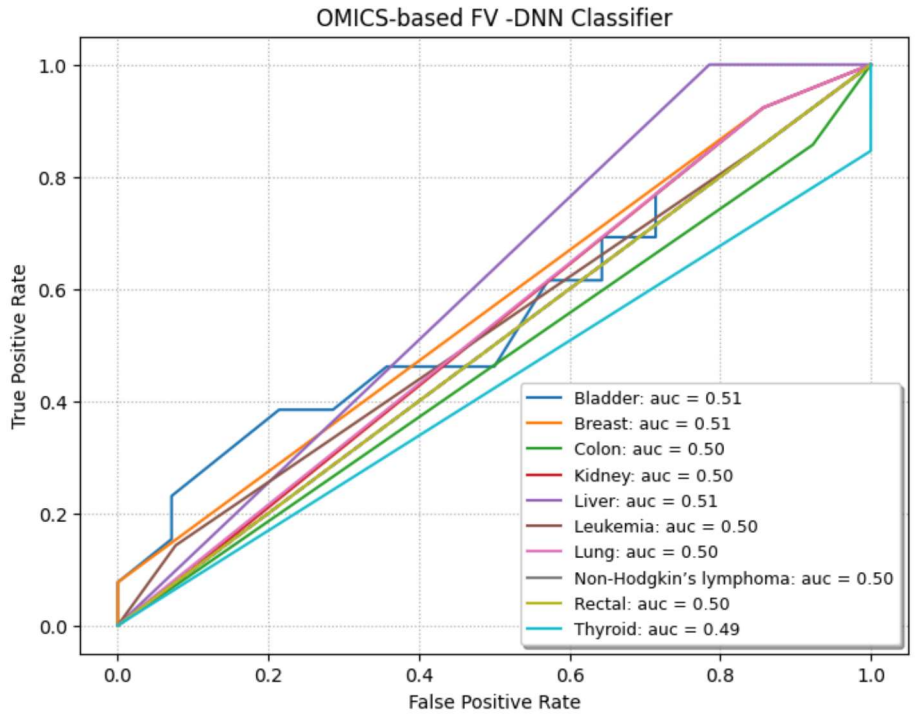
```
Epoch 16/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 17/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 18/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 19/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 20/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
1/1 [==============================] - 0s 30ms/step
------------------------------------------------------
*** Working with Fold 10 :***
Epoch 1/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 2/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 3/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 4/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 5/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 6/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 7/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.4333
Epoch 8/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 9/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 10/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.3833
Epoch 11/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.4583
Epoch 12/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.4750
Epoch 13/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.4833
Epoch 14/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.4583
Epoch 15/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.4833
Epoch 16/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.4500
Epoch 17/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.4667
Epoch 18/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 19/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 20/20
15/15 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000
1/1 [==============================] - 0s 44ms/step
------------------------------------------------------
Results using test data: AUC = 0.4923
```

```
# plot the roc curve for the omics models
pyplot.figure(figsize=[8, 6])
pyplot.grid( which='major', axis='both',linestyle=':')
pyplot.plot([0, 1], [0, 1], color='navy', lw=1.5, linestyle='dashed')
pyplot.plot(DL_omCancerFPR[0], DL_omCancerTPR[0],label='Bladder: auc = %0.2f' % np.array(DL_omCancerAUC[0]).mean())
pyplot.plot(DL_omCancerFPR[1], DL_omCancerTPR[1],label='Breast: auc = %0.2f' % np.array(DL_omCancerAUC[1]).mean())
pyplot.plot(DL_omCancerFPR[2], DL_omCancerTPR[2],label='Colon: auc = %0.2f' % np.array(DL_omCancerAUC[2]).mean())
pyplot.plot(DL_omCancerFPR[3], DL_omCancerTPR[3],label='Kidney: auc = %0.2f' % np.array(DL_omCancerAUC[3]).mean())
pyplot.plot(DL_omCancerFPR[4], DL_omCancerTPR[4],label='Liver: auc = %0.2f' % np.array(DL_omCancerAUC[4]).mean())
pyplot.plot(DL_omCancerFPR[5], DL_omCancerTPR[5],label='Leukemia: auc = %0.2f' % np.array(DL_omCancerAUC[5]).mean())
pyplot.plot(DL_omCancerFPR[6], DL_omCancerTPR[6],label='Lung: auc = %0.2f' % np.array(DL_omCancerAUC[6]).mean())
pyplot.plot(DL_omCancerFPR[7], DL_omCancerTPR[7],label='Non-Hodgkin's lymphoma: auc = %0.2f' % np.array(DL_omCancerAUC[7]).m
pyplot.plot(DL_omCancerFPR[8], DL_omCancerTPR[8],label='Rectal: auc = %0.2f' % np.array(DL_omCancerAUC[8]).mean())
pyplot.plot(DL_omCancerFPR[9], DL_omCancerTPR[9],label='Thyroid: auc = %0.2f' % np.array(DL_omCancerAUC[9]).mean())

# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.title('OMICS-based FV -DNN Classifier')

# show the legend
pyplot.legend(loc='best',fontsize = 9,shadow=True)
# pyplot.savefig('Figures/OMICS_FV_DNN_10Cancers.jpeg')
```

```
# show the plot
pyplot.show()
```




```
## Define the DL Model
DL_model = Sequential()
DL_model.add(Dense(64, input_dim=550, kernel_initializer='normal', activation='relu'))
# DL_model.add(Dropout(0.2))
DL_model.add(Dense(32,  activation='relu', kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
DL_model.add(Dense(1, activation='sigmoid'))
skf = StratifiedKFold(n_splits=10, shuffle = True, random_state = 10)


cancertypes = ["Bladder","Breast", "Colon", "Kidney", "Liver","Luk",
               "Lung", "NonHodgkinLymph", "Rectal", "Thyroid" ]

DL_emCancerAUC = []
DL_emCancerFPR= []
DL_emCancerTPR= []

for cancer in cancertypes:
    print('\n*Working with '+ cancer + " Cancer*" )

 ## Read  combined genes (positive and negative genes) per Cancer
    #Genes = pd.read_csv('Datasets/combinedGenes_cancer/'+str(cancer)+'_Genes.csv')
    Bladder=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Bladder_Genes.csv")
    Breast=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Breast_Genes.csv")
    Colon=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Colon_Genes.csv")
    Kidney=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Kidney_Genes.csv")
    Liver=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Liver_Genes.csv")
    Luk=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Luk_Genes.csv")
    Lung=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Lung_Genes.csv")
    NonHodgkinLymph=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/NonHodgkinLymph_Genes.csv")
    Rectal=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Rectal_Genes.csv")
    Thyroid=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Thyroid_Genes.csv")


    ## Read generated BERT embeddings for each genes
    #Embed = np.genfromtxt('BERT_Embed/'+str(cancer)+'_Embed.txt')
    Bladder_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Bladder_Embed.txt", usecols=range(679
    Breast_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Breast_Embed.txt", usecols=range(422))
    Colon_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Colon_Embed.txt", usecols=range(411))
    Kidney_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Kidney_Embed.txt", usecols=range(413))
    Liver_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Liver_Embed.txt", usecols=range(694))
    Luk_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Luk_Embed.txt", usecols=range(270))
```

```
    Luk_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Luk_Embed.txt", usecols=range(270))
    Lung_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Lung_Embed.txt", usecols=range(545))
    NonHodgkinLymph_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/NonHodgkinLymph_Embed.txt", u
    Rectal_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Rectal_Embed.txt", usecols=range(686))
    Thyroid_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Thyroid_Embed.txt", usecols=range(556

    #Embed_df = pd.DataFrame(Embed)
    Bladder_Embed_df = pd.DataFrame(Bladder_Embed)
    Breast_Embed_df = pd.DataFrame(Breast_Embed)
    Colon_Embed_df = pd.DataFrame(Colon_Embed)
    Kidney_Embed_df = pd.DataFrame(Kidney_Embed)
    Liver_Embed_df = pd.DataFrame(Liver_Embed)
    Luk_Embed_df = pd.DataFrame(Luk_Embed)
    Lung_Embed_df = pd.DataFrame(Lung_Embed)
    NonHodgkinLymph_Embed_df = pd.DataFrame(NonHodgkinLymph_Embed)
    Rectal_Embed_df = pd.DataFrame(Rectal_Embed)
    Thyroid_Embed_df = pd.DataFrame(Thyroid_Embed)

    ## The embeddings have the same order of the combined genes,
    #  so concat them using the index
    #allG_em = pd.merge(Genes, Embed_df,right_index=True,left_index=True)
    # ...


    Bladder_allG_em = pd.concat([Bladder, Bladder_Embed_df], axis=1)
    Breast_allG_em = pd.concat([Breast, Breast_Embed_df], axis=1)
    Colon_allG_em = pd.concat([Colon, Colon_Embed_df], axis=1)
    Kidney_allG_em = pd.concat([Kidney, Kidney_Embed_df], axis=1)
    Liver_allG_em = pd.concat([Liver, Liver_Embed_df], axis=1)
    Luk_allG_em = pd.concat([Luk, Luk_Embed_df], axis=1)
    Lung_allG_em = pd.concat([Lung, Lung_Embed_df], axis=1)
    NonHodgkinLymph_allG_em = pd.concat([NonHodgkinLymph, NonHodgkinLymph_Embed_df], axis=1)
    Rectal_allG_em = pd.concat([Rectal, Rectal_Embed_df], axis=1)
    Thyroid_allG_em = pd.concat([Thyroid, Thyroid_Embed_df], axis=1)

# ...

    ## Read gene mutation and gene expression
    #GE = pd.read_csv('Datasets/OMICS/'+str(cancer)+'_gene_exp.csv')
    Bladder_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Bladder_gene_exp.csv")
    Breast_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Breast_gene_exp.csv")
    Colon_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Colon_gene_exp.csv")
    Kidney_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Kidney_gene_exp.csv")
    Liver_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Liver_gene_exp.csv")
    Luk_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Luk_gene_exp.csv")
    Lung_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Lung_gene_exp.csv")
    NonHodgkinLymph_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/NonHodgkinLymph_gene_exp.csv")
    Rectal_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Rectal_gene_exp.csv")
    Thyroid_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Thyroid_gene_exp.csv")

    #Mu = pd.read_csv('Datasets/OMICS/'+str(cancer)+'_gene_mut.csv')
    Bladder_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Bladder_gene_mut.csv")
    Breast_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Breast_gene_mut.csv")
    Colon_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Colon_gene_mut.csv")
    Kidney_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Kidney_gene_mut.csv")
    Liver_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Liver_gene_mut.csv")
    Luk_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Luk_gene_mut.csv")
    Lung_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Lung_gene_mut.csv")
    NonHodgkinLymph_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/NonHodgkinLymph_gene_mut.csv")
    Rectal_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Rectal_gene_mut.csv")
    Thyroid_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Thyroid_gene_mut.csv")

    # merge OMICS features and remove the samples GE values

    #OMICS = pd.merge(GE,MU, on=['Gene'], how='inner')
    Bladder_gene_OMICS = pd.concat([Bladder_gene_exp, Bladder_gene_mut], axis=1)
    Breast_gene_OMICS = pd.concat([Breast_gene_exp, Breast_gene_mut], axis=1)
    Colon_gene_OMICS = pd.concat([Colon_gene_exp, Colon_gene_mut], axis=1)
    Kidney_gene_OMICS = pd.concat([Kidney_gene_exp, Kidney_gene_mut], axis=1)
    Liver_gene_OMICS = pd.concat([Liver_gene_exp, Liver_gene_mut], axis=1)
    Luk_gene_OMICS = pd.concat([Luk_gene_exp, Luk_gene_mut], axis=1)
```

```
Luk_gene_OMICS = pd.concat([Luk_gene_exp, Luk_gene_mut], axis=1)
Lung_gene_OMICS = pd.concat([Lung_gene_exp, Lung_gene_mut], axis=1)
NonHodgkinLymph_gene_OMICS = pd.concat([NonHodgkinLymph_gene_exp, NonHodgkinLymph_gene_mut], axis=1)
Rectal_gene_OMICS = pd.concat([Rectal_gene_exp, Rectal_gene_mut], axis=1)
Thyroid_gene_OMICS = pd.concat([Thyroid_gene_exp, Thyroid_gene_mut], axis=1)

#CncerOM = OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Bladder_CncerOM = Bladder_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Breast_CncerOM = Breast_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Colon_CncerOM = Colon_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Kidney_CncerOM = Kidney_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Liver_CncerOM = Liver_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Luk_CncerOM = Luk_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Lung_CncerOM = Lung_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
NonHodgkinLymph_CncerOM = NonHodgkinLymph_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Rectal_CncerOM = Rectal_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Thyroid_CncerOM = Thyroid_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]



# integrate all festures and get the lable, remove all genes with no label
#CancerFV = pd.merge(allG_em, CncerOM, on=['Gene'], how='outer')
Bladder_CancerFV = pd.concat([Bladder_allG_em, Bladder_CncerOM],axis=1)
Breast_CancerFV = pd.concat([Breast_allG_em, Breast_CncerOM],axis=1)
Colon_CancerFV = pd.concat([Colon_allG_em, Colon_CncerOM], axis=1)
Kidney_CancerFV = pd.concat([Kidney_allG_em, Kidney_CncerOM], axis=1)
Liver_CancerFV = pd.concat([Liver_allG_em, Liver_CncerOM], axis=1)
Luk_CancerFV = pd.concat([Luk_allG_em, Luk_CncerOM], axis=1)
Lung_CancerFV = pd.concat([Lung_allG_em, Lung_CncerOM], axis=1)
NonHodgkinLymph_CancerFV = pd.concat([NonHodgkinLymph_allG_em, NonHodgkinLymph_CncerOM], axis=1)
Rectal_CancerFV = pd.concat([Rectal_allG_em, Rectal_CncerOM], axis=1)
Thyroid_CancerFV = pd.concat([Thyroid_allG_em, Thyroid_CncerOM], axis=1)

# CancerFV = CancerFV[CancerFV['Label'].notna()]
Bladder_CancerFV = Bladder_CancerFV[Bladder_CancerFV['Label'].notna()]
Breast_CancerFV = Breast_CancerFV[Breast_CancerFV['Label'].notna()]
Colon_CancerFV = Colon_CancerFV[Colon_CancerFV['Label'].notna()]
Kidney_CancerFV = Kidney_CancerFV[Kidney_CancerFV['Label'].notna()]
Liver_CancerFV = Liver_CancerFV[Liver_CancerFV['Label'].notna()]
Luk_CancerFV = Luk_CancerFV[Luk_CancerFV['Label'].notna()]
Lung_CancerFV = Lung_CancerFV[Lung_CancerFV['Label'].notna()]
NonHodgkinLymph_CancerFV = NonHodgkinLymph_CancerFV[NonHodgkinLymph_CancerFV['Label'].notna()]
Rectal_CancerFV = Rectal_CancerFV[Rectal_CancerFV['Label'].notna()]
Thyroid_CancerFV = Thyroid_CancerFV[Thyroid_CancerFV['Label'].notna()]


#CancerFV = CancerFV.fillna(0)
Bladder_CancerFV = Bladder_CancerFV.fillna(0)
Breast_CancerFV = Breast_CancerFV.fillna(0)
Colon_CancerFV = Colon_CancerFV.fillna(0)
Kidney_CancerFV = Kidney_CancerFV.fillna(0)
Liver_CancerFV = Liver_CancerFV.fillna(0)
Luk_CancerFV = Luk_CancerFV.fillna(0)
Lung_CancerFV = Lung_CancerFV.fillna(0)
NonHodgkinLymph_CancerFV = NonHodgkinLymph_CancerFV.fillna(0)
Rectal_CancerFV = Rectal_CancerFV.fillna(0)
Thyroid_CancerFV = Thyroid_CancerFV.fillna(0)

Bladder_FV = Bladder_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Breast_FV = Breast_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Colon_FV = Colon_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Kidney_FV = Kidney_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Liver_FV = Liver_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Luk_FV = Luk_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Lung_FV = Lung_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
NonHodgkinLymph_FV = NonHodgkinLymph_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Rectal_FV = Rectal_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)
Thyroid_FV = Thyroid_CancerFV.drop(["Gene", "Sequence","Label","MAX","AVG","MED","MIN","COUNT"], axis=1)

X = np.array(Bladder_FV)
X = np.array(Breast_FV)
```

```
X = np.array(Breast_FV)
X = np.array(Colon_FV)
X = np.array(Kidney_FV)
X = np.array(Liver_FV)
X = np.array(Luk_FV)
X = np.array(Lung_FV)
X = np.array(NonHodgkinLymph_FV)
X = np.array(Rectal_FV)
X = np.array(Thyroid_FV)

#Y = np.array(CancerFV['Label'],dtype=int)
Y = np.array(Bladder_CancerFV['Label'],dtype=int)
Y = np.array(Breast_CancerFV['Label'],dtype=int)
Y = np.array(Colon_CancerFV['Label'],dtype=int)
Y = np.array(Kidney_CancerFV['Label'],dtype=int)
Y = np.array(Liver_CancerFV['Label'],dtype=int)
Y = np.array(Luk_CancerFV['Label'],dtype=int)
Y = np.array(Lung_CancerFV['Label'],dtype=int)
Y = np.array(NonHodgkinLymph_CancerFV['Label'],dtype=int)
Y = np.array(Rectal_CancerFV['Label'],dtype=int)
Y = np.array(Thyroid_CancerFV['Label'],dtype=int)

print('Features and Label shapes', X.shape, Y.shape)
print('++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++')

# all evaluation lists
auc = []
fpr = []
tpr = []

# Compile model
DL_model.compile(loss='hinge', optimizer='adam', metrics=['accuracy'])

foldCounter = 1
# Start training and testing
for train_index, test_index in  skf.split(X,Y):

    print("*** Working with Fold %i :***" %foldCounter)
    #Apply normalization using MaxAbsolute normlization
    max_abs_scaler = MinMaxScaler()
    X_train = max_abs_scaler.fit(X[train_index])
    X_train_transform = max_abs_scaler.transform(X[train_index])
    X_test_transform = max_abs_scaler.transform(X[test_index])

    DL_model.fit(X_train_transform, Y[train_index], epochs=20,batch_size=16)
    predictedScore = DL_model.predict(X_test_transform)
    #  our model's predictions.
    predictedClass= np.argmax(predictedScore, axis=0)

    #print("@@ Validation and evaluation of fold %i @@" %foldCounter)
    fr, tr, _ = roc_curve(Y[test_index], predictedScore)
    fpr.append(fr)
    tpr.append(tr)
    #print("AUC =  %f" %roc_auc_score(Y[test_index], predictedScore))
    auc.append(roc_auc_score(Y[test_index], predictedScore))

    print('------------------------------------------------------')
    foldCounter += 1

#------------------------------------------------------------
Fpr = np.array(sorted(list(itertools.chain.from_iterable(fpr))))
Tpr = np.array(sorted(list(itertools.chain.from_iterable(tpr))))
#------------------------------------------------
### Print Evaluation Metrics.....................
print("Results:AUC = " + str( np.array(auc).mean().round(decimals=4) ))

DL_aucROC = np.array(auc).mean().round(decimals=4)
DL_emCancerAUC.append(np.array(DL_aucROC).mean())
DL_emCancerFPR.append(Fpr)
DL_emCancerTPR.append(Tpr)
```

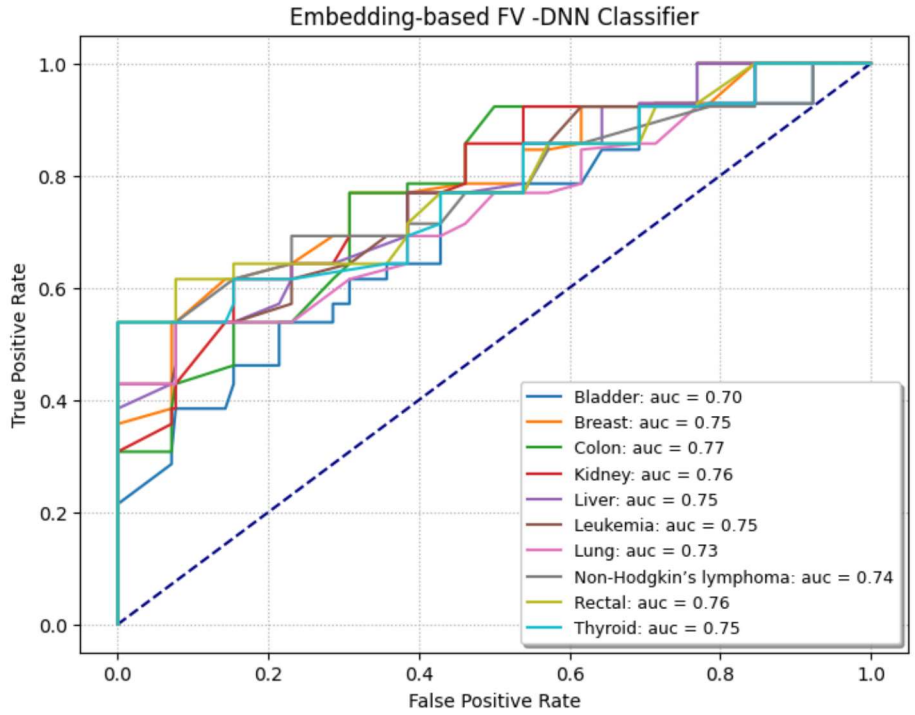
```
Epoch 15/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6981 - accuracy: 0.8042
Epoch 16/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6980 - accuracy: 0.8042
Epoch 17/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6980 - accuracy: 0.8042
Epoch 18/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6979 - accuracy: 0.8042
Epoch 19/20
15/15 [==============================] - 0s 5ms/step - loss: 0.6978 - accuracy: 0.8042
Epoch 20/20
15/15 [==============================] - 0s 4ms/step - loss: 0.6977 - accuracy: 0.8042
1/1 [==============================] - 0s 30ms/step
------------------------------------------------------
*** Working with Fold 10 :***
Epoch 1/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7104 - accuracy: 0.7917
Epoch 2/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7101 - accuracy: 0.7917
Epoch 3/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7100 - accuracy: 0.7917
Epoch 4/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7100 - accuracy: 0.7917
Epoch 5/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7099 - accuracy: 0.7917
Epoch 6/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7099 - accuracy: 0.7917
Epoch 7/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7098 - accuracy: 0.7917
Epoch 8/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7098 - accuracy: 0.7917
Epoch 9/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7097 - accuracy: 0.7917
Epoch 10/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7097 - accuracy: 0.7917
Epoch 11/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7097 - accuracy: 0.7917
Epoch 12/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7096 - accuracy: 0.7917
Epoch 13/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7096 - accuracy: 0.7917
Epoch 14/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7096 - accuracy: 0.7917
Epoch 15/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7095 - accuracy: 0.7917
Epoch 16/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7095 - accuracy: 0.7917
Epoch 17/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7095 - accuracy: 0.7917
Epoch 18/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7095 - accuracy: 0.7917
Epoch 19/20
15/15 [==============================] - 0s 4ms/step - loss: 0.7095 - accuracy: 0.7917
Epoch 20/20
15/15 [==============================] - 0s 3ms/step - loss: 0.7095 - accuracy: 0.7917
1/1 [==============================] - 0s 32ms/step
------------------------------------------------------
Results:AUC = 0.7537
```

```
# plot the roc curve for the model
pyplot.figure(figsize=[8, 6])
pyplot.plot([0, 1], [0, 1], color='navy', lw=1.5, linestyle='dashed')
pyplot.grid( which='major', axis='both',linestyle=':')
pyplot.plot(DL_emCancerFPR[0], DL_emCancerTPR[0],label='Bladder: auc = %0.2f' % np.array(DL_emCancerAUC[0]).mean())
pyplot.plot(DL_emCancerFPR[1], DL_emCancerTPR[1],label='Breast: auc = %0.2f' % np.array(DL_emCancerAUC[1]).mean())
pyplot.plot(DL_emCancerFPR[2], DL_emCancerTPR[2],label='Colon: auc = %0.2f' % np.array(DL_emCancerAUC[2]).mean())
pyplot.plot(DL_emCancerFPR[3], DL_emCancerTPR[3],label='Kidney: auc = %0.2f' % np.array(DL_emCancerAUC[3]).mean())
pyplot.plot(DL_emCancerFPR[4], DL_emCancerTPR[4],label='Liver: auc = %0.2f' % np.array(DL_emCancerAUC[4]).mean())
pyplot.plot(DL_emCancerFPR[5], DL_emCancerTPR[5],label='Leukemia: auc = %0.2f' % np.array(DL_emCancerAUC[5]).mean())
pyplot.plot(DL_emCancerFPR[6], DL_emCancerTPR[6],label='Lung: auc = %0.2f' % np.array(DL_emCancerAUC[6]).mean())
pyplot.plot(DL_emCancerFPR[7], DL_emCancerTPR[7],label='Non-Hodgkin's lymphoma: auc = %0.2f' % np.array(DL_emCancerAUC[7]).m
pyplot.plot(DL_emCancerFPR[8], DL_emCancerTPR[8],label='Rectal: auc = %0.2f' % np.array(DL_emCancerAUC[8]).mean())
pyplot.plot(DL_emCancerFPR[9], DL_emCancerTPR[9],label='Thyroid: auc = %0.2f' % np.array(DL_emCancerAUC[9]).mean())

# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.title('Embedding-based FV -DNN Classifier')

# show the legend
```

```
# show the legend
pyplot.legend(loc='best',fontsize = 9,shadow=True)
# pyplot.legend(bbox_to_anchor=(1.05, 1.0, 0.1, 0.1), loc='best',fontsize = 9)
#pyplot.savefig('Figures/Embeddings_FV_DNN_10Cancers.jpeg')
# show the plot
pyplot.show()
```




```
## OncoRTT-DL OMICS-based FV
## Define the Classifiers

DL_model = Sequential()
DL_model.add(Dense(64, input_dim=555, kernel_initializer='normal', activation='tanh'))
# DL_model.add(Dropout(0.2))
DL_model.add(Dense(32,  activation='tanh', kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
DL_model.add(Dense(1, activation='sigmoid'))


DL_CancerAUC = []
DL_CancerFPR= []
DL_CancerTPR= []

for cancer in cancertypes:
    print('\n*Working with '+ cancer + " Cancer*" )

    ## Read  combined genes (positive and negative geBnes) per Cancer
    Bladder=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Bladder_Genes.csv")
    Breast=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Breast_Genes.csv")
    Colon=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Colon_Genes.csv")
    Kidney=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Kidney_Genes.csv")
    Liver=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Liver_Genes.csv")
    Luk=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Luk_Genes.csv")
    Lung=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Lung_Genes.csv")
    NonHodgkinLymph=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/NonHodgkinLymph_Genes.csv")
    Rectal=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Rectal_Genes.csv")
    Thyroid=pd.read_csv("/content/drive/MyDrive/deep files/Combined genes/Thyroid_Genes.csv")


    ## Read generated BERT embeddings for each genes
    #Embed = np.genfromtxt('BERT_Embed/'+str(cancer)+'_Embed.txt')
    Bladder_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Bladder_Embed.txt",usecols=range(679))
    Breast_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Breast_Embed.txt", usecols=range(422))
    Colon_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Colon_Embed.txt", usecols=range(411))
    Kidney_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Kidney_Embed.txt", usecols=range(413))
    Liver_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Liver_Embed.txt", usecols=range(694))
```

```
    Luk_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Luk_Embed.txt", usecols=range(270))
    Lung_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Lung_Embed.txt", usecols=range(545))
    NonHodgkinLymph_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/NonHodgkinLymph_Embed.txt", u
    Rectal_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Rectal_Embed.txt", usecols=range(686))
    Thyroid_Embed=np.genfromtxt("/content/drive/MyDrive/deep files/Embedded text files/Thyroid_Embed.txt", usecols=range(556

    #Embed_df = pd.DataFrame(Embed)
    Bladder_Embed_df = pd.DataFrame(Bladder_Embed)
    Breast_Embed_df = pd.DataFrame(Breast_Embed)
    Colon_Embed_df = pd.DataFrame(Colon_Embed)
    Kidney_Embed_df = pd.DataFrame(Kidney_Embed)
    Liver_Embed_df = pd.DataFrame(Liver_Embed)
    Luk_Embed_df = pd.DataFrame(Luk_Embed)
    Lung_Embed_df = pd.DataFrame(Lung_Embed)
    NonHodgkinLymph_Embed_df = pd.DataFrame(NonHodgkinLymph_Embed)
    Rectal_Embed_df = pd.DataFrame(Rectal_Embed)
    Thyroid_Embed_df = pd.DataFrame(Thyroid_Embed)

    ## The embeddings have the same order of the combined genes,
    #  so concat them using the index
    #allG_em = pd.merge(Genes, Embed_df,right_index=True,left_index=True)
    # ...


    Bladder_allG_em = pd.concat([Bladder, Bladder_Embed_df], axis=1)
    Breast_allG_em = pd.concat([Breast, Breast_Embed_df], axis=1)
    Colon_allG_em = pd.concat([Colon, Colon_Embed_df], axis=1)
    Kidney_allG_em = pd.concat([Kidney, Kidney_Embed_df], axis=1)
    Liver_allG_em = pd.concat([Liver, Liver_Embed_df], axis=1)
    Luk_allG_em = pd.concat([Luk, Luk_Embed_df], axis=1)
    Lung_allG_em = pd.concat([Lung, Lung_Embed_df], axis=1)
    NonHodgkinLymph_allG_em = pd.concat([NonHodgkinLymph, NonHodgkinLymph_Embed_df], axis=1)
    Rectal_allG_em = pd.concat([Rectal, Rectal_Embed_df], axis=1)
    Thyroid_allG_em = pd.concat([Thyroid, Thyroid_Embed_df], axis=1)

# ...

    ## Read gene mutation and gene expression
    #GE = pd.read_csv('Datasets/OMICS/'+str(cancer)+'_gene_exp.csv')
    Bladder_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Bladder_gene_exp.csv")
    Breast_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Breast_gene_exp.csv")
    Colon_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Colon_gene_exp.csv")
    Kidney_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Kidney_gene_exp.csv")
    Liver_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Liver_gene_exp.csv")
    Luk_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Luk_gene_exp.csv")
    Lung_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Lung_gene_exp.csv")
    NonHodgkinLymph_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/NonHodgkinLymph_gene_exp.csv")
    Rectal_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Rectal_gene_exp.csv")
    Thyroid_gene_exp=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Thyroid_gene_exp.csv")

    #Mu = pd.read_csv('Datasets/OMICS/'+str(cancer)+'_gene_mut.csv')
    Bladder_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Bladder_gene_mut.csv")
    Breast_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Breast_gene_mut.csv")
    Colon_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Colon_gene_mut.csv")
    Kidney_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Kidney_gene_mut.csv")
    Liver_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Liver_gene_mut.csv")
    Luk_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Luk_gene_mut.csv")
    Lung_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Lung_gene_mut.csv")
    NonHodgkinLymph_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/NonHodgkinLymph_gene_mut.csv")
    Rectal_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Rectal_gene_mut.csv")
    Thyroid_gene_mut=pd.read_csv("/content/drive/MyDrive/deep files/OMICS/Thyroid_gene_mut.csv")

    # merge OMICS features and remove the samples GE values

    #OMICS = pd.merge(GE,MU, on=['Gene'], how='inner')
    Bladder_gene_OMICS = pd.concat([Bladder_gene_exp, Bladder_gene_mut], axis=1)
    Breast_gene_OMICS = pd.concat([Breast_gene_exp, Breast_gene_mut], axis=1)
    Colon_gene_OMICS = pd.concat([Colon_gene_exp, Colon_gene_mut], axis=1)
    Kidney_gene_OMICS = pd.concat([Kidney_gene_exp, Kidney_gene_mut], axis=1)
    Liver_gene_OMICS = pd.concat([Liver_gene_exp, Liver_gene_mut], axis=1)
```

```
Luk_gene_OMICS = pd.concat([Luk_gene_exp, Luk_gene_mut], axis=1)
Lung_gene_OMICS = pd.concat([Lung_gene_exp, Lung_gene_mut], axis=1)
NonHodgkinLymph_gene_OMICS = pd.concat([NonHodgkinLymph_gene_exp, NonHodgkinLymph_gene_mut], axis=1)
Rectal_gene_OMICS = pd.concat([Rectal_gene_exp, Rectal_gene_mut], axis=1)
Thyroid_gene_OMICS = pd.concat([Thyroid_gene_exp, Thyroid_gene_mut], axis=1)

#CncerOM = OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Bladder_CncerOM = Bladder_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Breast_CncerOM = Breast_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Colon_CncerOM = Colon_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Kidney_CncerOM = Kidney_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Liver_CncerOM = Liver_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Luk_CncerOM = Luk_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Lung_CncerOM = Lung_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
NonHodgkinLymph_CncerOM = NonHodgkinLymph_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Rectal_CncerOM = Rectal_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]
Thyroid_CncerOM = Thyroid_gene_OMICS[["Gene","MAX","AVG","MED","MIN","COUNT"]]



# integrate all festures and get the lable, remove all genes with no label
#CancerFV = pd.merge(allG_em, CncerOM, on=['Gene'], how='outer')
Bladder_CancerFV = pd.concat([Bladder_allG_em, Bladder_CncerOM],axis=1)
Breast_CancerFV = pd.concat([Breast_allG_em, Breast_CncerOM],axis=1)
Colon_CancerFV = pd.concat([Colon_allG_em, Colon_CncerOM], axis=1)
Kidney_CancerFV = pd.concat([Kidney_allG_em, Kidney_CncerOM], axis=1)
Liver_CancerFV = pd.concat([Liver_allG_em, Liver_CncerOM], axis=1)
Luk_CancerFV = pd.concat([Luk_allG_em, Luk_CncerOM], axis=1)
Lung_CancerFV = pd.concat([Lung_allG_em, Lung_CncerOM], axis=1)
NonHodgkinLymph_CancerFV = pd.concat([NonHodgkinLymph_allG_em, NonHodgkinLymph_CncerOM], axis=1)
Rectal_CancerFV = pd.concat([Rectal_allG_em, Rectal_CncerOM], axis=1)
Thyroid_CancerFV = pd.concat([Thyroid_allG_em, Thyroid_CncerOM], axis=1)

# CancerFV = CancerFV[CancerFV['Label'].notna()]
Bladder_CancerFV = Bladder_CancerFV[Bladder_CancerFV['Label'].notna()]
Breast_CancerFV = Breast_CancerFV[Breast_CancerFV['Label'].notna()]
Colon_CancerFV = Colon_CancerFV[Colon_CancerFV['Label'].notna()]
Kidney_CancerFV = Kidney_CancerFV[Kidney_CancerFV['Label'].notna()]
Liver_CancerFV = Liver_CancerFV[Liver_CancerFV['Label'].notna()]
Luk_CancerFV = Luk_CancerFV[Luk_CancerFV['Label'].notna()]
Lung_CancerFV = Lung_CancerFV[Lung_CancerFV['Label'].notna()]
NonHodgkinLymph_CancerFV = NonHodgkinLymph_CancerFV[NonHodgkinLymph_CancerFV['Label'].notna()]
Rectal_CancerFV = Rectal_CancerFV[Rectal_CancerFV['Label'].notna()]
Thyroid_CancerFV = Thyroid_CancerFV[Thyroid_CancerFV['Label'].notna()]


#CancerFV = CancerFV.fillna(0)
Bladder_CancerFV = Bladder_CancerFV.fillna(0)
Breast_CancerFV = Breast_CancerFV.fillna(0)
Colon_CancerFV = Colon_CancerFV.fillna(0)
Kidney_CancerFV = Kidney_CancerFV.fillna(0)
Liver_CancerFV = Liver_CancerFV.fillna(0)
Luk_CancerFV = Luk_CancerFV.fillna(0)
Lung_CancerFV = Lung_CancerFV.fillna(0)
NonHodgkinLymph_CancerFV = NonHodgkinLymph_CancerFV.fillna(0)
Rectal_CancerFV = Rectal_CancerFV.fillna(0)
Thyroid_CancerFV = Thyroid_CancerFV.fillna(0)

Bladder_FV = Bladder_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Breast_FV = Breast_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Colon_FV = Colon_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Kidney_FV = Kidney_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Liver_FV = Liver_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Luk_FV = Luk_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Lung_FV = Lung_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
NonHodgkinLymph_FV = NonHodgkinLymph_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Rectal_FV = Rectal_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)
Thyroid_FV = Thyroid_CancerFV.drop(["Gene", "Sequence","Label"], axis=1)

X = np.array(Bladder_FV)
```

```
X = np.array(Breast_FV)
X = np.array(Colon_FV)
X = np.array(Kidney_FV)
X = np.array(Liver_FV)
X = np.array(Luk_FV)
X = np.array(Lung_FV)
X = np.array(NonHodgkinLymph_FV)
X = np.array(Rectal_FV)
X = np.array(Thyroid_FV)

#Y = np.array(CancerFV['Label'],dtype=int)
Y = np.array(Bladder_CancerFV['Label'],dtype=int)
Y = np.array(Breast_CancerFV['Label'],dtype=int)
Y = np.array(Colon_CancerFV['Label'],dtype=int)
Y = np.array(Kidney_CancerFV['Label'],dtype=int)
Y = np.array(Liver_CancerFV['Label'],dtype=int)
Y = np.array(Luk_CancerFV['Label'],dtype=int)
Y = np.array(Lung_CancerFV['Label'],dtype=int)
Y = np.array(NonHodgkinLymph_CancerFV['Label'],dtype=int)
Y = np.array(Rectal_CancerFV['Label'],dtype=int)
Y = np.array(Thyroid_CancerFV['Label'],dtype=int)

print('Features and Label shapes', X.shape, Y.shape)
print('++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++')

# all evaluation lists
auc = []
fpr = []
tpr = []

# Compile model
DL_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

foldCounter = 1
# Start training and testing
for train_index, test_index in  skf.split(X,Y):

    print("*** Working with Fold %i :***" %foldCounter)
    #Apply normalization using MaxAbsolute normlization
    max_abs_scaler = MinMaxScaler()
    X_train = max_abs_scaler.fit(X[train_index])
    X_train_transform = max_abs_scaler.transform(X[train_index])
    X_test_transform = max_abs_scaler.transform(X[test_index])

    DL_model.fit(X_train_transform, Y[train_index], epochs=20,
             batch_size=16)
    predictedScore = DL_model.predict(X_test_transform)
    #  our model's predictions.
    predictedClass= np.argmax(predictedScore, axis=0)

    #print("@@ Validation and evaluation of fold %i @@" %foldCounter)
    fr, tr, _ = roc_curve(Y[test_index], predictedScore)
    fpr.append(fr)
    tpr.append(tr)
    #print("AUC =  %f" %roc_auc_score(Y[test_index], predictedScore))
    auc.append(roc_auc_score(Y[test_index], predictedScore))

    print('------------------------------------------------------')
    foldCounter += 1

#-------------------------------------------------------
Fpr = np.array(sorted(list(itertools.chain.from_iterable(fpr))))
Tpr = np.array(sorted(list(itertools.chain.from_iterable(tpr))))
#-----------------------------------------------
### Print Evaluation Metrics.....................
print("Results:AUC = " + str( np.array(auc).mean().round(decimals=4) ))

DL_aucROC = np.array(auc).mean().round(decimals=4)
DL_CancerAUC.append(np.array(DL_aucROC).mean())
DL_CancerFPR.append(Fpr)
```

```
DL_CancerTPR.append(Tpr)
```

```
Epoch 15/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2796 - accuracy: 0.8292
Epoch 16/20
15/15 [==============================] - 0s 5ms/step - loss: 0.2782 - accuracy: 0.8292
Epoch 17/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2782 - accuracy: 0.8083
Epoch 18/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2786 - accuracy: 0.8292
Epoch 19/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2783 - accuracy: 0.8083
Epoch 20/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2770 - accuracy: 0.8292
1/1 [==============================] - 0s 28ms/step
------------------------------------------------------
*** Working with Fold 10 :***
Epoch 1/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2800 - accuracy: 0.7875
Epoch 2/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2819 - accuracy: 0.8000
Epoch 3/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2783 - accuracy: 0.8000
Epoch 4/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2822 - accuracy: 0.7958
Epoch 5/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2889 - accuracy: 0.8167
Epoch 6/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2807 - accuracy: 0.8083
Epoch 7/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2831 - accuracy: 0.8042
Epoch 8/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2849 - accuracy: 0.7833
Epoch 9/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2818 - accuracy: 0.7917
Epoch 10/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2794 - accuracy: 0.8000
Epoch 11/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2848 - accuracy: 0.8167
Epoch 12/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2837 - accuracy: 0.8042
Epoch 13/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2806 - accuracy: 0.7875
Epoch 14/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2810 - accuracy: 0.8167
Epoch 15/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2819 - accuracy: 0.8000
Epoch 16/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2792 - accuracy: 0.8042
Epoch 17/20
15/15 [==============================] - 0s 4ms/step - loss: 0.2780 - accuracy: 0.8167
Epoch 18/20
15/15 [==============================] - 0s 5ms/step - loss: 0.2865 - accuracy: 0.7750
Epoch 19/20
15/15 [==============================] - 0s 3ms/step - loss: 0.2887 - accuracy: 0.8000
Epoch 20/20
15/15 [==============================] - 0s 3ms/step - loss: 0.3016 - accuracy: 0.8208
1/1 [==============================] - 0s 32ms/step
------------------------------------------------------
Results:AUC = 0.9172
```

```
# plot the roc curve for the integrated FV models
pyplot.figure(figsize=[8, 6])

pyplot.plot(DL_CancerFPR[0], DL_CancerTPR[0],label='Bladder: auc = %0.2f' % np.array(DL_CancerAUC[0]).mean())
pyplot.plot(DL_CancerFPR[1], DL_CancerTPR[1],label='Breast: auc = %0.2f' % np.array(DL_CancerAUC[1]).mean())
pyplot.plot(DL_CancerFPR[2], DL_CancerTPR[2],label='Colon: auc = %0.2f' % np.array(DL_CancerAUC[2]).mean())
pyplot.plot(DL_CancerFPR[3], DL_CancerTPR[3],label='Kidney: auc = %0.2f' % np.array(DL_CancerAUC[3]).mean())
pyplot.plot(DL_CancerFPR[4], DL_CancerTPR[4],label='Liver: auc = %0.2f' % np.array(DL_CancerAUC[4]).mean())
pyplot.plot(DL_CancerFPR[5], DL_CancerTPR[5],label='Leukemia: auc = %0.2f' % np.array(DL_CancerAUC[5]).mean())
pyplot.plot(DL_CancerFPR[6], DL_CancerTPR[6],label='Lung: auc = %0.2f' % np.array(DL_CancerAUC[6]).mean())
pyplot.plot(DL_CancerFPR[7], DL_CancerTPR[7],label='Non-Hodgkin's lymphoma: auc = %0.2f' % np.array(DL_CancerAUC[7]).mean())
pyplot.plot(DL_CancerFPR[8], DL_CancerTPR[8],label='Rectal: auc = %0.2f' % np.array(DL_CancerAUC[8]).mean())
pyplot.plot(DL_CancerFPR[9], DL_CancerTPR[9],label='Thyroid: auc = %0.2f' % np.array(DL_CancerAUC[9]).mean())
pyplot.plot([0, 1], [0, 1], color='navy', lw=1.5, linestyle='dashed')

# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.title('Integrated FV(OMICS+Embeddings)-DNN Classifier')
```

```
pyplot.grid( which='major', axis='both',linestyle=':')

# show the legend
pyplot.legend(loc='best',fontsize = 9,shadow=True)
# pyplot.savefig('Figures/Integrated_FV_DNN_10Cancers.jpeg')
# show the plot
pyplot.show()
```