CO19322: 2019-2023 Batch

```
!pip install py-entitymatching #match entities between two tables using supervised learning
```

```
  Downloading py_entitymatching-0.4.0.tar.gz (2.0 MB)
     |████████████████████████████████| 2.0 MB 5.3 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (2.23.0)
Requirement already satisfied: ipython>=5.6 in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (7.9.0)
Requirement already satisfied: matplotlib>=2.2.4 in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (3.2.2)
Collecting PyPrind
  Downloading PyPrind-2.11.3-py2.py3-none-any.whl (8.4 kB)
Collecting py-stringsimjoin>=0.3.0
  Downloading py_stringsimjoin-0.3.2.tar.gz (1.1 MB)
     |████████████████████████████████| 1.1 MB 52.0 MB/s
Requirement already satisfied: cloudpickle>=0.2.1 in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (1.5.0)
Requirement already satisfied: pyparsing>=2.1.4 in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (3.0.9)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (1.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (1.7.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from py-entitymatching) (1.21.6)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entityma
Requirement already satisfied: pexpect in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entitymatching) (4.8.0)
Requirement already satisfied: pygments in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entitymatching) (2.6.1)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entitymatching) (5.7.1
Requirement already satisfied: pickleshare in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entitymatching) (0.7.5)
Requirement already satisfied: backcall in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entitymatching) (0.2.0)
Collecting jedi>=0.10
  Downloading jedi-0.18.2-py2.py3-none-any.whl (1.6 MB)
     |████████████████████████████████| 1.6 MB 27.8 MB/s
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entitymatching) (57.
Requirement already satisfied: decorator in /usr/local/lib/python3.8/dist-packages (from ipython>=5.6->py-entitymatching) (4.4.2)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.8/dist-packages (from jedi>=0.10->ipython>=5.6->py-entit
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.2.4->py-entitymatching
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.2.4->py-entitymatch
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.2.4->py-entitymatching) (0.
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.8/dist-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=5.6-
Requirement already satisfied: wcwidth in /usr/local/lib/python3.8/dist-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=5.6->py
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from py-stringsimjoin>=0.3.0->py-entitymatching) (1.
Requirement already satisfied: pandas>=0.16.0 in /usr/local/lib/python3.8/dist-packages (from py-stringsimjoin>=0.3.0->py-entitymatch
Collecting py_stringmatching>=0.2.1
  Downloading py_stringmatching-0.4.2.tar.gz (661 kB)
     |████████████████████████████████| 661 kB 72.4 MB/s
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=0.16.0->py-stringsimjoin>=0.3.0->
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn>=0.22->py-entitymatc
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.8/dist-packages (from pexpect->ipython>=5.6->py-entitymatchi
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->py-entitymatching) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->py-e
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->py-entitymatching) (2022.
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->py-entitymatching) (3.0.4)
Building wheels for collected packages: py-entitymatching, py-stringsimjoin, py-stringmatching
  Building wheel for py-entitymatching (setup.py) ... done
  Created wheel for py-entitymatching: filename=py_entitymatching-0.4.0-cp38-cp38-linux_x86_64.whl size=2650981 sha256=90190cecdabb3e
  Stored in directory: /root/.cache/pip/wheels/cb/e7/dc/2b33e80855a70e58bbc60b31d3a5f22a93a3a3ef606eb50594
  Building wheel for py-stringsimjoin (setup.py) ... done
  Created wheel for py-stringsimjoin: filename=py_stringsimjoin-0.3.2-cp38-cp38-linux_x86_64.whl size=4117987 sha256=7a2aca0e6bea9ac7
  Stored in directory: /root/.cache/pip/wheels/7f/61/96/0aa1d87a2d0a9329ea415ffaf74c875c9344434844177f2b62
  Building wheel for py-stringmatching (setup.py) ... done
  Created wheel for py-stringmatching: filename=py_stringmatching-0.4.2-cp38-cp38-linux_x86_64.whl size=2191994 sha256=8167978491fd8f
  Stored in directory: /root/.cache/pip/wheels/6d/01/57/0131e0a87fa6471ec265bd9777de2e789a4bd1248c279fa875
Successfully built py-entitymatching py-stringsimjoin py-stringmatching
Installing collected packages: PyPrind, py-stringmatching, jedi, py-stringsimjoin, py-entitymatching
Successfully installed PyPrind-2.11.3 jedi-0.18.2 py-entitymatching-0.4.0 py-stringmatching-0.4.2 py-stringsimjoin-0.3.2
```

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

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

```
# Importing the libraries
import sys
import pandas as pd
import os
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
path_variable = "/content/gdrive/MyDrive/1 Dataset/Music"
method = "/content/gdrive/MyDrive/1 Dataset/weights_music"


import py_entitymatching as em #match entities between two tables using supervised learning
# Display the versions
print('python version: ' + sys.version )
print('pandas version: ' + pd.__version__ ) #data science/data analysis and machine learning tasks
print('magellan version: ' + em.__version__ ) #Magellan provides how-to guides that tell users what to do in each EM scenario
```

```
python version: 3.8.16 (default, Dec  7 2022, 01:12:13)
[GCC 7.5.0]
pandas version: 1.3.5
magellan version: 0.4.0
```

```
# Load csv files as dataframes and set the key attribute in the dataframe
path_A = '/content/gdrive/MyDrive/1 Dataset/Music/music1.csv'
path_B = '/content/gdrive/MyDrive/1 Dataset/Music/music2.csv'
A = em.read_csv_metadata(path_A, key='Sno')
B = em.read_csv_metadata(path_B, key='Sno')

# Display number of tuples in the datasets
print('Number of tuples in A: ' + str(len(A)))
print('Number of tuples in B: ' + str(len(B)))
print('Number of tuples in A X B (i.e the cartesian product): ' + str(len(A)*len(B)))
```

```
WARNING:py_entitymatching.io.parsers:Metadata file is not present in the given path; proceeding to read the csv file.
WARNING:py_entitymatching.io.parsers:Metadata file is not present in the given path; proceeding to read the csv file.
Number of tuples in A: 6907
Number of tuples in B: 55923
Number of tuples in A X B (i.e the cartesian product): 386260161
```

```
# verify values in S

d1 = dict()
d2 = dict()

t1 = A.to_numpy()
t2 = B.to_numpy()

for x in t1:
  if(x[-1] in d1):
    d1[x[-1]]+=1
  else:
    d1[x[-1]]=1

for x in t2:
  if(x[-1] in d2):
    d2[x[-1]]+=1
  else:
    d2[x[-1]]=1

print("d1 : ",d1)
print("d2 : ",d2)
```

```
d1 :  {'Country': 1367, 'Pop': 1758, 'Other': 538, 'Dance': 952, 'Alternative': 484, 'Soundtrack': 397, 'Electronic': 211, 'Rock': 515,
d2 :  {'Dance': 8274, 'Other': 13135, 'Electronic': 4019, 'Soundtrack': 3594, 'Country': 8317, 'Rock': 3248, 'Pop': 7130, 'Alternative':
```

```
import pickle #serializing and deserializing a Python object structure
import numpy as np #working with arrays
import torch #Torch is an open source ML library used for creating deep neural networks

dbfile = open("/content/gdrive/MyDrive/1 Dataset/Music/files/feature_matrix", 'rb')
cand_file = open("/content/gdrive/MyDrive/1 Dataset/Music/files/C", 'rb')
S = pickle.load(dbfile) #serializing and de-serializing a Python object structure
cand = pickle.load(cand_file)

S = torch.tensor(S.values,dtype=torch.double) #multi-dimensional matrix containing elements of a single data type
S = S[:,3:]
# S = S.to_numpy(dtype=np.longdouble)
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

# Initial weights would be 1/(num_attributes*num_sim_scores) due to add-to-1 constraint
w_init  = (1 / S.shape[1] )
# W = np.full((S.shape[0], S.shape[1]), w_init,dtype=np.longdouble)
W = torch.DoubleTensor(S.shape[1],1).fill_(w_init) #multi-dimensional matrix containing elements of a single data type.
```

```
W = torch.DoubleTensor(S.shape[1],1).fill_(w_init) #multi dimensional matrix containing elements of a single data type.


# ith index of genre_groups denotes genre of ith record pair
genre_groups = [] #used to split the data into groups based on some criteria
for i in cand['ltable_Group']:
    genre_groups.append(i) # appends an element to the end of the list


# ith row of RLS matrix will return RLS value of the ith record pair
# return nx1 matrix, where n=number of record pairs
def get_RLS(W,S):# Recursive Least Square #Returns the URL to the storage location of a data file or directory tracked in a DVC project
    rls = torch.matmul(S,torch.square(W)) #performs matrix multiplications if both arguments are 2D and computes their dot product if both ar
    return rls
    # return torch.diag(rls)

def fair_calc(weights,S,flag,k): #function

  rls_matrix = get_RLS(weights,S)
  cand['score'] = rls_matrix.numpy()
  candy = cand.sort_values(by=['score'], ascending=False) #sorts a data frame in Ascending or Descending order of passed Column

  d3 = dict()
  d4 = dict()

  temp = candy.to_numpy()
  cnt = 0

  for _id,pair in enumerate(temp):
    if(_id>k):

      break

    if(pair[11] in d3):
      d3[pair[11]]+=1
    else:
      d3[pair[11]]=1

    if(pair[20] in d4):
      d4[pair[20]]+=1
    else:
      d4[pair[20]]=1

    if(pair[11]==(pair[20])):
      cnt+=1
  mini = -1
  maxi = 1
  d_skew = {}
  flagy = 0
  for d in d1.keys():
    if(d in d3.keys()):
      d_skew[d] = np.log( d3[d]*len(t1)/(d1[d]*k))

      if(flagy==0):
        flagy+=1
        mini = d_skew[d]
        maxi = d_skew[d]

      mini = min(mini,d_skew[d])
      maxi = max(maxi,d_skew[d])

  if(flag):
    print("Group distribution in result : ",d3)
    print("Skew metrix : ", maxi-mini)

  return maxi-mini

def get_precomputed_values(W,S,phi,tau):
    numerator = torch.exp(phi*get_RLS(W,S)) - torch.exp(phi* (tau-get_RLS(W,S)))
    # print(numerator)
    denominator = torch.exp(phi*get_RLS(W,S)) + torch.exp(phi* (tau-get_RLS(W,S)))
    return torch.divide(numerator,denominator)

def derivative_loss_function(W,S,j,precomputed_values):
    m = torch.matmul(torch.transpose(S[:,j].reshape(-1,1),0,1),precomputed_values)
    der = 2 * W[j,0] * m
    return der
```

```
def algorithm_gradient_descent(W,S,learning_rate,iterations,phi,tau,flag_tweak,k):

    for i in range(iterations):
        W_new = torch.zeros(W.shape) #Returns a tensor filled with the scalar value 0 , with the shape defined by the variable argument size
        sum_W_new = 0
        precomputed_values_em = get_precomputed_values(W,S,phi,tau)
        for j in range(W.shape[0]):
            loss_value = derivative_loss_function(W,S,j,precomputed_values_em)
            W_new[j,0] = W[j,0] - learning_rate * loss_value
            sum_W_new += W_new[j,0]
        for j in range(W.shape[0]):
            W_new[j,0] = W_new[j,0] / float(sum_W_new)
        W = W_new.double()
        if((i+1)%10==0):
          print("Iteration " + str(i+1) + " completed ")

        if(flag_tweak):
          f_val = fair_calc(W,S,False,k)
          epsi = 0.1

          for j in range(W.shape[0]):
            W[j,0] += epsi
            f1 = fair_calc(W,S,False,k)

            W[j,0] -= 2*epsi
            f2 = fair_calc(W,S,False,k)
            # print(f1,f2,f_val)
            if(f_val<=f1 and f_val<=f2):
              # print("val")
              W[j,0] += epsi

            elif(f1<=f2 and f1<=f_val):
              # print("f1")
              W[j,0] += 2*epsi

    return W

#No learning
print(W)
fair_calc(W,S,True,500)
```

```
 tensor([[0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357],
         [0.0357]], dtype=torch.float64)
 Group distribution in result :  {'Alternative': 111, 'Dance': 35, 'Country': 246, 'Rock': 79, 'Other': 5, 'Hip-Hop': 7, 'Pop': 12, 'Elec
 Skew metrix :  3.514470723052824
 3.514470723052824
```

```
#No tweak of weights
weights1 = algorithm_gradient_descent(W,S,1e-5,100,-100,1,False,500) # gradient descent algorithm is an approximate and iterative method for
```

```
 Iteration 10 completed
 Iteration 20 completed
```

```
Iteration 30 completed
Iteration 40 completed
Iteration 50 completed
Iteration 60 completed
Iteration 70 completed
Iteration 80 completed
Iteration 90 completed
Iteration 100 completed
```

```
print(weights1)
fair_calc(weights1,S,True,500)
```

```
tensor([[6.8612e-05],
        [3.3806e-06],
        [9.0364e-10],
        [2.6130e-06],
        [1.5958e-06],
        [1.4624e-06],
        [3.2026e-09],
        [2.6926e-07],
        [4.0319e-02],
        [9.5047e-01],
        [8.5094e-08],
        [8.6403e-03],
        [4.1154e-04],
        [6.3659e-05],
        [5.8570e-09],
        [8.7798e-06],
        [2.1046e-06],
        [8.9070e-08],
        [3.0561e-06],
        [9.8269e-08],
        [1.1543e-08],
        [3.1129e-09],
        [4.3995e-08],
        [8.4605e-10],
        [2.4957e-07],
        [2.2925e-10],
        [6.5277e-09],
        [1.5063e-11]], dtype=torch.float64)
Group distribution in result :  {'Country': 364, 'Rock': 11, 'Pop': 51, 'Other': 39, 'Dance': 36}
Skew metrix :  2.523051658778157
2.523051658778157
```

```
#Tweak weights
weights2 = algorithm_gradient_descent(W,S,1e-5,100,-100,1,True,500)
```

```
Iteration 10 completed
Iteration 20 completed
Iteration 30 completed
Iteration 40 completed
Iteration 50 completed
Iteration 60 completed
Iteration 70 completed
Iteration 80 completed
Iteration 90 completed
Iteration 100 completed
```

```
print(weights2)
fair_calc(weights2,S,True,500)
```

```
tensor([[ 0.0979],
        [ 0.0996],
        [ 0.1040],
        [ 0.1004],
        [ 0.1063],
        [ 0.1059],
        [ 0.0960],
        [ 0.1007],
        [-0.8705],
        [ 1.1412],
        [ 0.1923],
        [ 0.6219],
        [ 0.0408],
        [-0.0311],
        [ 0.1035],
        [-0.0018],
        [ 0.1035],
        [ 0.0192],
        [ 0.0181],
        [ 0.0191],
```

```
               [ 0.1119],
               [ 0.0894],
               [-0.0396],
               [ 0.1048],
               [ 0.0542],
               [-0.0143],
               [ 0.2311],
               [ 0.0954]], dtype=torch.float64)
      Group distribution in result :  {'Other': 40, 'Country': 299, 'Rock': 11, 'Pop': 51, 'Alternative': 100}
      Skew metrix :  2.3263413645321025
      2.3263413645321025
```

```
def comp_overlap(w1,w2,k):
  rls_matrix = get_RLS(w1,S)
  cand['score'] = rls_matrix.numpy()
  candy1 = cand.sort_values(by=['score'], ascending=False)

  # print(candy)

  dw1 = dict()
  temp1 = candy1.to_numpy()

  rls_matrix = get_RLS(w2,S)
  cand['score'] = rls_matrix.numpy()
  candy2 = cand.sort_values(by=['score'], ascending=False)

  # print(candy)
  dw2 = dict()
  temp2 = candy2.to_numpy()
  overlp = 0
  dname = {}


  for _i,i in enumerate(temp1):
    if(_i>k-1):
      break
    dname[str(i[1])+"#"+str(i[2])] = 1

  for _j,j in enumerate(temp2):
      if(_j>k-1):
        break
      if(str(j[1])+"#"+str(j[2]) in  dname.keys()):
        overlp+=1

  return overlp



i=0
w_list = []

for k in [250,500,750,1000]:
  w_list.append(algorithm_gradient_descent(W,S,1e-5,100,-100,1,True,k)) #appends an element to the end of the list
  i+=1
```

```
     Iteration 10 completed
     Iteration 20 completed
     Iteration 30 completed
     Iteration 40 completed
     Iteration 50 completed
     Iteration 60 completed
     Iteration 70 completed
     Iteration 80 completed
     Iteration 90 completed
     Iteration 100 completed
     Iteration 10 completed
     Iteration 20 completed
     Iteration 30 completed
     Iteration 40 completed
     Iteration 50 completed
     Iteration 60 completed
     Iteration 70 completed
     Iteration 80 completed
     Iteration 90 completed
     Iteration 100 completed
     Iteration 10 completed
     Iteration 20 completed
     Iteration 30 completed
     Iteration 40 completed
     Iteration 50 completed
```

```
Iteration 60 completed
Iteration 70 completed
Iteration 80 completed
Iteration 90 completed
Iteration 100 completed
Iteration 10 completed
Iteration 20 completed
Iteration 30 completed
Iteration 40 completed
Iteration 50 completed
Iteration 60 completed
Iteration 70 completed
Iteration 80 completed
Iteration 90 completed
Iteration 100 completed
```

```
weight_file = open(method+'weights_list', 'wb')
pickle.dump(w_list, weight_file) #to store the object data to the file
weight_file.close()
```

```
weight_file = open(method+'weights1', 'wb')
pickle.dump(weights1, weight_file)
weight_file.close()

weight_file = open(method+'weights2', 'wb')
pickle.dump(weights2, weight_file)
weight_file.close()
```

```
weight_file = open(method+'weights1', 'rb')
weights1 = pickle.load(weight_file)

weight_file = open(method+'weights2', 'rb')
weights2 = pickle.load(weight_file)
```

```
l = get_RLS(weights2,S)
se = set()
print(l.shape)
for i in range(l.shape[0]):
    se.add(float(l[i,0]))
print(se)
print(len(se))
```

```
torch.Size([12829, 1])
{0.15722333010601375, 1.2170469871510008, 0.3683803241299728, 1.2227987126681263, 1.0348041878863512, 0.7730500730782772, 1.272390805353
4668
```

```
path_G = '/content/gdrive/MyDrive/1 Dataset/Music/music_labeled_data.csv'
G = em.read_csv_metadata(path_G,
                         key='_id',
                         ltable=A, rtable=B,
                         fk_ltable='ltable.Sno', fk_rtable='rtable.Sno')
print(len(G))

gdic = {}
cnt_no_g = 0
cnt_g = 0

G1 = G.to_numpy()
for i,x in enumerate(G1):
  s = str(x[2])+"#"+str(x[3])
  s1 = str(x[3])+"#"+str(x[2])
  # print(s)
  gdic[s] = x[-3]
  gdic[s1] = x[-3]
  if(x[-3]==1):
    cnt_g+=1
  else:
    cnt_no_g+=1
print("Positive in ground truth",cnt_g)
print("Negative in ground truth",cnt_no_g)

print(gdic.keys())

def check_gt(w1,S,k):
```

```
  rls_matrix = get_RLS(w1,S)
  cand['score'] = rls_matrix.numpy()
  candy1 = cand.sort_values(by=['score'], ascending=False)
  temp1 = candy1.to_numpy()

  overlp = 0
  neg = 0

  for _j,j in enumerate(temp1):
      if(_j>k-1):
        break
      s = str(j[1])+"#"+str(j[2])

      if(s in  gdic.keys()):
        # print(s)
        if(gdic[s]==1):
          overlp+=1
        else:
          neg+=1

  return overlp,neg
```

```
WARNING:py_entitymatching.io.parsers:Metadata file is not present in the given path; proceeding to read the csv file.
539
Positive in ground truth 132
Negative in ground truth 407
dict_keys(['111#53124', '53124#111', '148#50767', '50767#148', '206#41214', '41214#206', '211#19812', '19812#211', '250#53111', '53111#2
```

```
weight_file = open('/content/gdrive/MyDrive/1 Dataset/weights_music/weights_list', 'rb')
w_list = pickle.load(weight_file) # serializing and deserializing a Python object structure

i=0
for k in [250,500,750,1000]:
  print("Top-",k)
  weight_this = w_list[i]
  i+=1
  # print("Weights =",weight_this)
  print("Fairness skew for w1 " + " is "+ str(fair_calc(weights1,S,False,k)))
  print("Fairness skew for w" + str(k)+ " is "+ str(fair_calc(weight_this,S,False,k)))
  print("Overlap with base result = "+str(comp_overlap(weights1,weight_this,k))+" /"+ str(k))
  print("Overlap with ground truth weights1", check_gt(weights1,S,k))
  print("Overlap with ground truth weights_this", check_gt(weight_this,S,k),"\n")
```

```
Top- 250
Fairness skew for w1  is 3.156967876781878
Fairness skew for w250 is 2.234057838008778
Overlap with base result = 98 /250
Overlap with ground truth weights1 (3, 0)
Overlap with ground truth weights_this (5, 0)

Top- 500
Fairness skew for w1  is 2.523051658778157
Fairness skew for w500 is 2.3263413645321025
Overlap with base result = 363 /500
Overlap with ground truth weights1 (5, 0)
Overlap with ground truth weights_this (5, 0)

Top- 750
Fairness skew for w1  is 2.873940665649845
Fairness skew for w750 is 1.3243853445213443
Overlap with base result = 154 /750
Overlap with ground truth weights1 (7, 0)
Overlap with ground truth weights_this (10, 0)

Top- 1000
Fairness skew for w1  is 2.302715134119735
Fairness skew for w1000 is 1.9170626780132305
Overlap with base result = 240 /1000
Overlap with ground truth weights1 (7, 0)
Overlap with ground truth weights_this (18, 0)
```

```
def get_results_em(weights):
  # Load the pre-labeled data
  rls_scores_G = get_RLS(weights,S)

  # make seperate column for predicted label
```

```
    count_pred = 0
    G['pred_label'] = 0
    for i in range(rls_scores_G.shape[0]):
        if float(rls_scores_G[i,0]) > 0.2:
            G['pred_label'][i] = 1
            count_pred+=1
        else:
            G['pred_label'][i] = 0
    print("\n")
    print("percentage of labels=1: " , count_pred/rls_scores_G.shape[0])

    print()
    print("--------------Evaluation summary---------------")
    # Evaluate the predictions
    eval_result = em.eval_matches(G, 'label', 'pred_label')
    em.print_eval_summary(eval_result)


print(W)
get_results_em(W)
```

```
tensor([[0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357],
        [0.0357]], dtype=torch.float64)
<ipython-input-32-895f59ee69f8>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  G['pred_label'][i] = 0


percentage of labels=1:  0.0

--------------Evaluation summary---------------
Precision : 0.0% (0/0)
Recall : 0.0% (0/132)
F1 : 0.0%
False positives : 0 (out of 0 positive predictions)
False negatives : 132 (out of 539 negative predictions)
```

```
get_results_em(weights1)
```

```
<ipython-input-32-895f59ee69f8>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  G['pred_label'][i] = 0
<ipython-input-32-895f59ee69f8>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  G['pred_label'][i] = 1


percentage of labels=1:  0.10647751188713071
```

```
--------------Evaluation summary---------------
Precision : 5.0% (1/20)
Recall : 0.76% (1/132)
F1 : 1.32%
False positives : 19 (out of 20 positive predictions)
False negatives : 131 (out of 519 negative predictions)
```

```
get_results_em(weights2)
```

```
<ipython-input-32-895f59ee69f8>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  G['pred_label'][i] = 1
<ipython-input-32-895f59ee69f8>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  G['pred_label'][i] = 0


percentage of labels=1:  0.7813547431600281

--------------Evaluation summary---------------
Precision : 21.53% (73/339)
Recall : 55.3% (73/132)
F1 : 31.0%
False positives : 266 (out of 339 positive predictions)
False negatives : 59 (out of 200 negative predictions)
```

✓ 10s completed at 4:48 AM