In [1]:

```
import networkx as nx
import torch
import numpy as np
import pandas as pd
import random
import community as community_louvain
from sklearn.model_selection import train_test_split
from itertools import product

# allCategories = pd.read_csv("categories.csv", index_col=0)

class Feedforward(torch.nn.Module):
    def __init__(self, input_size, hidden_size):
        super(Feedforward, self).__init__()

        self.input_size = input_size
        self.hidden_size  = hidden_size

        self.fc1 = torch.nn.Linear(self.input_size, self.hidden_size)
        self.fc2 = torch.nn.Linear(self.hidden_size, self.hidden_size)
        self.fc3 = torch.nn.Linear(self.hidden_size, self.hidden_size)
        self.fc4 = torch.nn.Linear(self.hidden_size, 1)

        self.relu = torch.nn.ReLU()
        self.out_act = torch.nn.Sigmoid()

    def forward(self, x):
        output = self.fc1(x)
        output = self.relu(output)

        output = self.fc2(output)
        output = self.relu(output)

        output = self.fc3(output)
        output = self.relu(output)

        output = self.fc4(output)
        output = self.out_act(output)

        return output
```

In [2]:

```
#Network to evaluate links
GMissingEdges = nx.read_gml("GraphMissingEdges.gml")

# To local test ----------------------------------------
# Remove 20% das arestas
proportion_edges = 0.2
edge_subset = random.sample(GMissingEdges.edges(), int(proportion_edges * GMi

# Cria uma cópia do grafo e remove arestas
GMissingEdgesTrain = GMissingEdges.copy()
GMissingEdgesTrain.remove_edges_from(edge_subset)
# print(edge_subset)
# To local test ----------------------------------------

dfGraphEdges = pd.DataFrame.from_dict(dict(GMissingEdgesTrain.edges()), orier
dfGraphEdges = dfGraphEdges.reset_index()
# Renomeia campos como index e edge, em comparação aos nós que estão conectad
dfGraphEdges = dfGraphEdges.rename(columns={'level_0': 'idx', 'level_1':'edge
print(dfGraphEdges)
```

```
print(dfGraphEdges.index)
print(dfGraphEdges.columns)
```

```
                          idx                    edge  weight
0      --DaPTJW3-tB1vP-PfdTEg  EL-iUP2pr6aJE2ZRVyNwyA       1
1      --DaPTJW3-tB1vP-PfdTEg  MhiBpIBNTCAm1Xd3WzRzjQ       1
2      --DaPTJW3-tB1vP-PfdTEg  dT70QOjn-o9pkdSAAPdSWQ       1
3      --SrzpvFLwP_YFwB_Cetow  4PINzgssH9dDbw36jofi_Q       1
4      --SrzpvFLwP_YFwB_Cetow  9FPs1mXHZEoEWo3kw9cwGQ       1
...                       ...                    ...     ...
42387  zzUj3ej4vm_DtvRxNvWDEw  tJcpzXzykNSLuzWwa1JQUw       1
42388  zzUj3ej4vm_DtvRxNvWDEw  trzuDWvJqEIxtqjsKHCrhg       1
42389  zzf3RkMI1Y2E1QaZqeU8yA  mZRKH9ngRY92bI_irrHq6w       1
42390  zzvlwkcNR1CCqOPXwuvz2A  4Jscimulh38Rq2hOgjb2Hg       1
42391  zzvlwkcNR1CCqOPXwuvz2A  _xAJZOKBMPOe47p1MphB2w       1

[42392 rows x 3 columns]
RangeIndex(start=0, stop=42392, step=1)
Index(['idx', 'edge', 'weight'], dtype='object')
```

In [3]:

```
# Create test file using data removed
dfToEvaluate = pd.DataFrame(edge_subset)
dfToEvaluate = dfToEvaluate.rename(columns={0:'venue1',1:'venue2'})
dfToEvaluate.to_csv("edgesToEvaluateTest.csv", index_label='linkID')
dfToEvaluate
```

Out[3]:

	venue1	venue2
0	jniApOOS8ppUHhESL7OzTg	KYYUvIJi7laFzK1NsaE77Q
1	gyFYZV4b_9TxG1ulQNi0Ig	SJr6Hs_XS4ubUq8NojqXzA
2	KVUOj74lBgogrdKcNQH_zQ	0sPOBQHlVvuhO1h-1p1ccQ
3	3HwoIoJcV7yDi3RV7T-oDQ	czzjfPe7kO9VxoYV9l-fxA
4	84JCu-4LvE6SDAglrJztGA	pTbkdBDDKxNVjKUZ_6RAug
...	...	...
10593	Pz13Ru_-U4qoZiu89ezVmQ	JOoblYsQjFT-47tkt6om0A
10594	e41TP5cXZqSrz50xCBJqZw	MzEH3h8meWt7fW146U7y0g
10595	PyIF7dcDPNpfmi8ks-sKOQ	GxzEsd-81hVP6C2h6wMvBw
10596	_T8qy9XAKAFLJdmoLg1Q-g	QePLHlU8MFaU2Sf9dzoLTg
10597	fy6srT4KpbE7ICBLdypEuQ	2PCz_uVX7GOXtGHNXAPXhw

10598 rows × 2 columns

In [4]:

```
GMissingEdges.nodes(data=True)
# GMissingEdges.edges(data=True)

dfGraphNodes = pd.DataFrame.from_dict(dict(GMissingEdgesTrain.nodes(data=True
dfGraphNodes.drop(['name','categories'],axis=1,inplace=True)

# Calculate louvain communities
louvainPartition = community_louvain.best_partition(GMissingEdges)
dfLouvainPartition = pd.DataFrame.from_dict(louvainPartition, orient='index')
dfLouvainPartition = dfLouvainPartition.rename(columns={0: 'cluster'})

# Add Louvain clusterization
dfGraphNodes = pd.concat([dfGraphNodes, dfLouvainPartition], axis=1)
t_index()
```

```
# print(dfGraphNodes, dfGraphEdges)
print(dfGraphNodes)
```

```
                       index  longitude   latitude stars reviewCount  cluste
r
0      DHCdMpffUncZWxaiYNHSZw -79.434315  43.646220   4.0           4
0
1      huCf4kwsoGl1YUHCjMJG5A -79.397848  43.631814   2.5           3
1
2      a6FJ9HcERvtGF4PYILF_fA -79.378986  43.654590   2.5          63
2
3      b8cwL5L3241tOcqXywEfLw -79.353691  43.683799   3.0           8
3
4      YQ_z9iDgdNjwJhZ-owHSjA -79.396689  43.674244   2.5           7
2
...                       ...        ...        ...   ...         ...
...
11075  q3bkTWv854XTLXq1F4pnKg -79.409645  43.645954   4.0          61
7
11076  8pGD3zt6HEL2xzaT3lqMFQ -79.408460  43.642893   4.5          12
2
11077  iByQmTmTdO7hP4n1grSSWQ -79.422871  43.662417   4.0          37
2
11078  lkM72Y21bjBqUGaW7iL7tQ -79.293509  43.803568   3.0          83
5
11079  vUef2kuyYWG7phLySoRJGw -79.357862  43.676379   4.0          22      10
9

[11080 rows x 6 columns]
```

A princípio tinha removido as colunas 'name' e 'categories', pois iria utilizar o atributo da aresta (weight) para tentar prever os links faltantes. Foi adicionado a clusterização de Louvain para tentar mais assertividade nos possíveis links, após esse cálculo a informação foi agregada aos atributos dos nós.

In [5]:

```
dfGraph = pd.merge(dfGraphNodes, dfGraphEdges, how="right", right_on=["idx"],
dfGraph.drop(['idx'],axis=1,inplace=True)
# dfGraph = dfGraph.set_index("index")

print(dfGraph)
print(dfGraph.info())
print(dfGraph.index)

# dfGraph.to_csv("dfGraph.csv")
```

```
                       index  longitude   latitude stars reviewCount  \
0      --DaPTJW3-tB1vP-PfdTEg -79.444674  43.677807   3.5          49
1      --DaPTJW3-tB1vP-PfdTEg -79.444674  43.677807   3.5          49
2      --DaPTJW3-tB1vP-PfdTEg -79.444674  43.677807   3.5          49
3      --SrzpvFLwP_YFwB_Cetow -79.288858  43.806750   3.5          43
4      --SrzpvFLwP_YFwB_Cetow -79.288858  43.806750   3.5          43
...                       ...        ...        ...   ...         ...
42387  zzUj3ej4vm_DtvRxNvWDEw -79.402828  43.643715   3.0         114
42388  zzUj3ej4vm_DtvRxNvWDEw -79.402828  43.643715   3.0         114
42389  zzf3RkMI1Y2E1QaZqeU8yA -79.370983  43.651883   4.5          33
42390  zzvlwkcNR1CCqOPXwuvz2A -79.393727  43.655822   3.5           7
42391  zzvlwkcNR1CCqOPXwuvz2A -79.393727  43.655822   3.5           7

       cluster                    edge  weight
0            2  EL-iUP2pr6aJE2ZRVyNwyA       1
1            2  MhiBpIBNTCAm1Xd3WzRzjQ       1
2            2  dT70QOjn-o9pkdSAAPdSWQ       1
3            7  4PINzgssH9dDbw36jofi_Q       1
4            7  9FPs1mXHZEoEWo3kw9cwGQ       1
                                  ...     ...
42387        4  tJcpzXzykNSLuzWwa1JQUw       1
```

```
42388        4  trzuDWvJqEIxtqjsKHCrhg       1
42389        7  mZRKH9ngRY92bI_irrHq6w       1
42390        5  4Jscimulh38Rq2hOgjb2Hg       1
42391        5  _xAJZOKBMPOe47p1MphB2w       1

[42392 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 42392 entries, 0 to 42391
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   index        42392 non-null  object
 1   longitude    42392 non-null  float64
 2   latitude     42392 non-null  float64
 3   stars        42392 non-null  object
 4   reviewCount  42392 non-null  object
 5   cluster      42392 non-null  int64
 6   edge         42392 non-null  object
 7   weight       42392 non-null  int64
dtypes: float64(2), int64(2), object(4)
memory usage: 2.9+ MB
None
Int64Index([    0,     1,     2,     3,     4,     5,     6,     7,     8,
                9,
            ...
            42382, 42383, 42384, 42385, 42386, 42387, 42388, 42389, 42390,
            42391],
           dtype='int64', length=42392)
```

Aqui os dataframes de arestas e nós foram mergeados, para capturar os dados de origem-destino da aresta juntamente com os atributos dos nós

In [6]:

```
dfEdgesToEvaluate = pd.read_csv('edgesToEvaluate.csv')
dfEdgesToEvaluate.drop(['linkID'], axis=1, inplace=True)
dfEdgesToEvaluate.rename(columns={'venue1': 'index', 'venue2':'edge'},inplace

# Generate zero cases to help test
combinedList = list(product(dfEdgesToEvaluate['index'], dfEdgesToEvaluate['ed
# print(combinedList)

dfT = pd.DataFrame(combinedList)
dfT.rename(columns={0: 'index', 1:'edge'},inplace=True)

dfGraph = pd.merge(dfT, dfGraph, how="outer", on=["index", "edge"])
dfGraph.drop(['longitude','latitude','stars','reviewCount','cluster', 'categ

# Remove edges to evaluate
dfGraph = pd.concat([dfGraph, dfEdgesToEvaluate])
dfGraph.drop_duplicates(subset=["index","edge"],keep=False, inplace=True)

dfGraph = pd.merge(dfGraph, dfGraphNodes, how="inner", on="index")

# weight = 0 and existEdge = False for created cases
dfGraph['weight'] = dfGraph['weight'].fillna(0)
dfGraph['existEdge'] = np.where((dfGraph.weight > 0 ), 'True', 'False')
# print(dfGraph.info())
print(dfGraph)

dfGraph = dfGraph.astype({"stars": float, "reviewCount": int, "existEdge": bo

dfGraph.to_csv("resultTest.csv")
```

```
                      index                    edge  weight  longitude  \
0      klu0zF1rWAoNAbKBsEvUog  oQFMJqDwNXbNMRbcmIYRYg     0.0 -79.363541
                              gLYFnycp8ktxMCvilFdLw     0.0 -79.363541
```

```
2       klu0zF1rWAoNAhKPsFyUog  Nxg73OigmRQQq0d1pKtkUQ     0.0 -79.363541
3       klu0zF1rWAoNAhKPsFyUog  hyXNS3tSmi6njhBjgo8eGw     0.0 -79.363541
4       klu0zF1rWAoNAhKPsFyUog  Sflaxtv6SR0lgbL7-pIGPQ     0.0 -79.363541
...                        ...                     ...     ...        ...
246770  zzUj3ej4vm_DtvRxNvWDEw  tJcpzXzykNSLuzWwa1JQUw     1.0 -79.402828
246771  zzUj3ej4vm_DtvRxNvWDEw  trzuDWvJqEIxtqjsKHCrhg     1.0 -79.402828
246772  zzf3RkMI1Y2E1QaZqeU8yA  mZRKH9ngRY92bI_irrHq6w     1.0 -79.370983
246773  zzvlwkcNR1CCqOPXwuvz2A  4Jscimulh38Rq2hOgjb2Hg     1.0 -79.393727
246774  zzvlwkcNR1CCqOPXwuvz2A  _xAJZOKBMPOe47p1MphB2w     1.0 -79.393727

         latitude stars reviewCount  cluster existEdge
0       43.709978   3.5         104        2     False
1       43.709978   3.5         104        2     False
2       43.709978   3.5         104        2     False
3       43.709978   3.5         104        2     False
4       43.709978   3.5         104        2     False
...           ...   ...         ...      ...       ...
246770  43.643715   3.0         114        4      True
246771  43.643715   3.0         114        4      True
246772  43.651883   4.5          33        7      True
246773  43.655822   3.5           7        5      True
246774  43.655822   3.5           7        5      True

[246775 rows x 9 columns]
```

Para incrementar mais os dados para treinamento, inclui algumas combinações de nós que não possuiam arestas para treinar também casos falsos em que não há arestas entre os nós, e na sequência para o dataset não ficar muito grande, foram deletadas várias arestas para voltar ao tamanho normal, pois o processamento estava inviável. Foi criado o campo para predição das arestas chamado 'existEdge'

In [7]:

```python
# Delete random ids because the structure is too heavy
print("Size of dataframe before deletion: %s" % (len(dfGraph)))
delSize = int(len(dfGraph)*0.8) # Come back to original size of dataframe +-

deletedItems = random.sample(range(len(dfGraph)), delSize)

dfGraph.drop(dfGraph.index[deletedItems], inplace=True)

print("Elements deleted %s" % (delSize))
print("Size of dataframe after deletion: %s" % (len(dfGraph)))

dfGraph = dfGraph.reset_index()
dfGraph.drop('level_0',axis=1,inplace=True)
# print (deletedItems)
print(dfGraph)
```

```
Size of dataframe before deletion: 246775
Elements deleted 197420
Size of dataframe after deletion: 49355
                        index                    edge  weight  longitude  \
0      klu0zF1rWAoNAhKPsFyUog  oQFMJqDwNXbNMRbcmIYRYg     0.0 -79.363541
1      klu0zF1rWAoNAhKPsFyUog  egLYFnycp8ktxMCvilFdLw     0.0 -79.363541
2      klu0zF1rWAoNAhKPsFyUog  Nxg73OigmRQQq0d1pKtkUQ     0.0 -79.363541
3      klu0zF1rWAoNAhKPsFyUog  pdTYUCGkYz35utxPyUMoag     0.0 -79.363541
4      klu0zF1rWAoNAhKPsFyUog  78Hx8KRI2SVKB-OibvEoag     0.0 -79.363541
...                       ...                     ...     ...        ...
49350  zwhgnF9ICofzzIAIuWtbkQ  bCc7Fi46nrgZHhx9f5gIGg     1.0 -79.456523
49351  zy_NHTqtfSrfTGGPoqy4Mw  zNL9Ajmn3gHUk__kpX7aIg     1.0 -79.448091
49352  zzUj3ej4vm_DtvRxNvWDEw  eSp5ge9VAwTywZKlJ_LBvA     1.0 -79.402828
49353  zzUj3ej4vm_DtvRxNvWDEw  o1FLiGssn5Wxc_POWSuZZA     1.0 -79.402828
49354  zzUj3ej4vm_DtvRxNvWDEw  tJcpzXzykNSLuzWwa1JQUw     1.0 -79.402828

                                   unt  cluster  existEdge
                                   104        2       True
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
1      43.709978    3.5          104        2       True
2      43.709978    3.5          104        2       True
3      43.709978    3.5          104        2       True
4      43.709978    3.5          104        2       True
...          ...    ...          ...      ...        ...
49350  43.670462    3.5           10        2       True
49351  43.644215    3.5           24       58       True
49352  43.643715    3.0          114        4       True
49353  43.643715    3.0          114        4       True
49354  43.643715    3.0          114        4       True

[49355 rows x 9 columns]
```

In [9]:

```
# Replace place ids to integers ids
placesId = dfGraphNodes['index'].to_dict()
# print(placesId)

# for index,place in placesId.items():
#     dfGraph.replace({'index':{place:index}},inplace=True)
#     dfGraph.replace({'edge':{place:index}},inplace=True)

dfGraph
```

Out[9]:

	index	edge	weight	longitude	latitude	stars	reviewCount	cluster	existEdge
0	2920	6729	0.0	-79.363541	43.709978	3.5	104	2	True
1	2920	6183	0.0	-79.363541	43.709978	3.5	104	2	True
2	2920	4031	0.0	-79.363541	43.709978	3.5	104	2	True
3	2920	9893	0.0	-79.363541	43.709978	3.5	104	2	True
4	2920	5752	0.0	-79.363541	43.709978	3.5	104	2	True
...	...	...	...	...	...	...	...	...	...
49350	2917	5852	1.0	-79.456523	43.670462	3.5	10	2	True
49351	2169	8515	1.0	-79.448091	43.644215	3.5	24	58	True
49352	175	9088	1.0	-79.402828	43.643715	3.0	114	4	True
49353	175	230	1.0	-79.402828	43.643715	3.0	114	4	True
49354	175	349	1.0	-79.402828	43.643715	3.0	114	4	True

49355 rows × 9 columns

Acima foi uma tentativa de usar os ids das ligações de nós (origem e destino) para a predição, ai transformei os ids dos nós em ids numéricos para o algoritmo tentar interpretar.

In [14]:

```
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Using {}".format(device))

focus = dfGraph['existEdge']
data = dfGraph.iloc[:,[3,4,5,6,7]]
data = data.astype({"stars": float, "reviewCount": int})


Y_tensor = torch.tensor(focus)
X_tensor = torch.tensor(data.to_numpy())
print(Y_tensor.shape)
print(X_tensor.shape)
```



```
= train_test_split(X_tensor, Y_tensor, test_
```

```
print("Training data:")
print(X_train.shape)
print(y_train.shape)

print("Test data:")
print(X_test.shape)
print(y_test.shape)

# Cast fields to float to avoid compatibility problems
X_train = X_train.float().to(device)
y_train = y_train.float().to(device)
X_test = X_test.float().to(device)
y_test = y_test.float().to(device)
```

```
Using cuda
torch.Size([49355])
torch.Size([49355, 5])
Training data:
torch.Size([39484, 5])
torch.Size([39484])
Test data:
torch.Size([9871, 5])
torch.Size([9871])
```

Transformação do dataframe em dados para o treinamento, aqui foram testadas várias combinações de features, mas nenhuma demonstrou ganho significante

In [15]:

```
#input = num_features (número de features), e hidden size = 20 (número de neu
num_features = X_train.shape[1]
print("num_features: "+str(num_features))

model = Feedforward(num_features, 20).to(device)
print(model)

criterion = torch.nn.MSELoss()
# criterion = torch.nn.BCEWithLogitsLoss()

# lr = Learning rate
optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)
# optimizer = torch.optim.SGD(model.parameters(), lr = 0.001)
```

```
num_features: 5
Feedforward(
  (fc1): Linear(in_features=5, out_features=20, bias=True)
  (fc2): Linear(in_features=20, out_features=20, bias=True)
  (fc3): Linear(in_features=20, out_features=20, bias=True)
  (fc4): Linear(in_features=20, out_features=1, bias=True)
  (relu): ReLU()
  (out_act): Sigmoid()
)
```

Foi testado outro algoritmo a não ser o MSELoss mas tbm sem ganhos significantes

In [16]:

```
model.eval()
y_pred = model(X_test)
before_train = criterion(y_pred, y_test)
print('Teste - perda antes do treinamento' , before_train.item())

model.train()
epoch = 15

# Zero gradients
```

```
    optimizer.zero_grad()

    # Forward pass
    y_pred = model(X_train)

    # Calculate loss
    loss = criterion(y_pred, y_train)

    print('Epoch {}: perda treino: {}'.format(epoch, loss.item()))

    # Backward pass
    loss.backward()
    # update pass
    optimizer.step()

model.eval()
y_pred = model(X_test)
after_train = criterion(y_pred, y_test)
print('Teste - perda depois do treinamento' , after_train.item())
```

```
Teste - perda antes do treinamento 0.0026627075858414173
Epoch 0: perda treino: 0.002692868933081627
Epoch 1: perda treino: 0.0012824563309550285
Epoch 2: perda treino: 0.000654550502076745
Epoch 3: perda treino: 0.0003609405248425901
Epoch 4: perda treino: 0.0002126830368069932
Epoch 5: perda treino: 0.00013247194874566048
Epoch 6: perda treino: 8.654539124108851e-05
Epoch 7: perda treino: 5.895810681977309e-05
Epoch 8: perda treino: 4.16621332988143e-05
Epoch 9: perda treino: 3.0403969503822736e-05
Epoch 10: perda treino: 2.2820693629910238e-05
Epoch 11: perda treino: 1.75602672243258e-05
Epoch 12: perda treino: 1.3814094018016476e-05
Epoch 13: perda treino: 1.1085472578997724e-05
Epoch 14: perda treino: 9.05930846784031e-06
Teste - perda depois do treinamento 7.511731382692233e-06
```

O treinamento da rede ao meu ver estava estranho, pois já iniciava com valor de perda extremamente baixo, depois aumentava um pouco e depois voltava ao inicial praticamente, por isso estava testando com poucas epocas.

In [17]:

```
# print(X_test)
# print(y_pred)
# print(after_train)

#Links to be evaluated
dfEdgesToEvaluate = pd.read_csv('edgesToEvaluateTest.csv')
dfEdgesToEvaluate = dfEdgesToEvaluate.rename(columns={'venue1': 'index', 'ver
dfEdgesToEvaluate.set_index('index',inplace=True)

# print(dfEdgesToEvaluate)
dfNodesToCopy = dfGraphNodes.copy()
dfNodesToCopy.set_index('index',inplace=True)
# print(dfNodesToCopy)

dfToTest = pd.merge(dfEdgesToEvaluate, dfNodesToCopy, on='index')
dfToTest.reset_index(inplace=True)
dfToTest.drop('linkID',axis=1,inplace=True)

# for index,place in placesId.items():
#     dfToTest.replace({'index':{place:index}},inplace=True)
#     dfToTest.replace({'index':{place:index}},inplace=True)
```

```
dfToTest['weight'] = 0

dfToTest.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10598 entries, 0 to 10597
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   index        10598 non-null  object 
 1   edge         10598 non-null  object 
 2   longitude    10598 non-null  float64
 3   latitude     10598 non-null  float64
 4   stars        10598 non-null  object 
 5   reviewCount  10598 non-null  object 
 6   cluster      10598 non-null  int64  
 7   weight       10598 non-null  int64  
dtypes: float64(2), int64(2), object(4)
memory usage: 662.5+ KB
```

Após treinamento, apliquei para validação nos dados fornecidos, porém infelizmente não obtive resultados expressivos

In [21]:

```
# focus = dfToTest['existEdge']
# Same column order
data = dfToTest.iloc[:,[2,3,4,5,6]]
data = data.astype({"stars": float, "reviewCount": int})

# print(focus)
# print(data.info())
# print(dfGraph.columns)

Y_tensor_eval = torch.zeros(len(dfEdgesToEvaluate),1)
X_tensor_eval = torch.tensor(data.to_numpy())

print("Test data:")
print(X_tensor_eval.shape)
print(Y_tensor_eval.shape)

# Cast fields to float to avoid compatibility problems
X_tensor_eval = X_tensor_eval.float().to(device)
Y_tensor_eval = Y_tensor_eval.float().to(device)
```

```
Test data:
torch.Size([10598, 5])
torch.Size([10598, 1])
```

In [22]:

```
model.eval()
y_pred = model(X_tensor_eval)
after_train = criterion(y_pred, Y_tensor_eval)
print('Teste - usando dados do treinamento' , after_train.item())

# print(len(y_pred))
# print(y_pred)

exitData = y_pred.detach().numpy()
print(len(exitData))
print(exitData)

binaryExit = np.where(exitData > 0, 1, 0)
print(len(binaryExit))
print(binaryExit)
```

```
# dfEdgesToEvaluate.set_index('linkID',inplace=True)
dfEdgesToEvaluate.drop(['edge'],axis=1,inplace=True,errors='ignore')

dfEdgesToEvaluate.to_csv("edgesToEvaluateAnswers.csv", columns=['linkID','lir
dfEdgesToEvaluate
```

```
Teste - usando dados do treinamento 0.996825098991394
10598
[[0.99718106]
 [0.99718106]
 [0.99718106]
 ...
 [0.9984682 ]
 [0.9996439 ]
 [0.99993277]]
10598
[[1]
 [1]
 [1]
 ...
 [1]
 [1]
 [1]]
```

Out[22]:

index	linkID	link
jniApOOS8ppUHhESL7OzTg	0	1
gyFYZV4b_9TxG1ulQNi0Ig	1	1
KVUOj74lBgogrdKcNQH_zQ	2	1
3HwoloJcV7yDi3RV7T-oDQ	3	1
84JCu-4LvE6SDAglrJztGA	4	1
...	...	...
Pz13Ru_-U4qoZiu89ezVmQ	10593	1
e41TP5cXZqSrz50xCBJqZw	10594	1
PyIF7dcDPNpfmi8ks-sKOQ	10595	1
_T8qy9XAKAFLJdmoLg1Q-g	10596	1
fy6srT4KpbE7ICBLdypEuQ	10597	1

10598 rows × 2 columns

Preparação do arquivo para upload na kaggle, não consegui identificar o motivo das respostas ficarem sempre 1 ou 0, sem meios termos