Response to Reviewer 3 Comments

Point 1: Authors failed to clearly explain the problem and their ANN based model. So authors should explain in detail what is the input to the model and outcome

Response 1: In order to establish an appropriate dataset for the training of the proposed neuronal network, a routine was developed in Python 3 in charge of normalizing the data obtained where basically a binarization of the CTX-M nucleotide sequences is carried out. The source code for this purpose is presented below.

In general terms, binarization consists of converting the nucleotides to their corresponding binary value by means of a function called vectorizeSequence (lines 7 to 11) and their correspondence to the binary value is established in lines 34 to 62, thus defining the input values (X) for the neural network, to generate the Y (CTX-M groups) the data are discriminated after the comma separator (,) in line 30.

All sequences are standardized to the value of the longest identified sequence and additional spaces are defined by the value N (lines 24 to 27).

The result is the file "dataGen.csv" where a total of 3896 values are generated for X and the 10 CTX-M groups (Classes).

The 10 most representative classes were selected in order to ensure a uniform distribution of classes for stratified cross validation in stage 2 (validation). Initially there were 17 classes from which only those with sequences represented at least 4 times within the test and validation data set were selected,

Each of the 10 classes corresponds to the following CTX-M groups respectively

Group CTX-M	Class
1.0	0
9.0	1
14.0	2
15.0	3
22.0	4
24.0	5
27.0	6
55.0	7
59.0	8
65.0	9

----------Code to pre-filter the ctx-m----------

```
#!/bin/bash
#software requerido:
module load software/bioinformatics/duk/2011
module load software/bioinformatics/emboss/6.6.0
file=$1
threads=$2
database=datafinalCTXM.fa
workdir=/home/dhceballosl/Projects/CTXM/
echo "executing: $0 $file $threads with database: $database"
#Para filtrar el metagenoma de referencia con la base de datos Consolidada de CTXM
rm -f duk_output_tmp
for k in 17 19 21 25 30 35 40 45 50 55 60 65
do
  mkdir duk_${k}_kmer_tmp
  duk -k ${k} -m duk_${k}_kmer_tmp/salida_duk.fastq ${database} ${file} >>
duk_output_tmp
done

#to process output file from duk
#Para procesar el archivo de salida de duk
better_kmer=`awk -F "||" -v kmer_r="" -v kmer_p="" -v reads="0" -v pvalue="0" -v
kmer_aux="" '{
  if($0 ~ /#Mer size:/){
    split($0,lista_mer,":");
    kmer_aux=lista_mer[2]+0;
  }
  if($0 ~ /# Total number of matched reads:/){
    split($0,lista_mer,":");
    if(reads <= (lista_mer[2]+0) ){
      kmer_r = kmer_aux+0;
      reads = lista_mer[2]+0
    }
  }
  if($0 ~ /#Avg number of Kmer for each read:/){
    split($0,lista_mer,":");
    if(pvalue <= (lista_mer[2]+0) ){
      kmer_p = kmer_aux+0;
      pvalue = lista_mer[2]+0
    }
  }
}END{
  if(kmer_r == kmer_p){
    print kmer_r
  }else{
    print kmer_p
  }
}' duk_output_tmp`
```

```
echo "the better kmer for duk is: $better_kmer"
mv duk_${better_kmer}_kmer_tmp duk_source_dir
rm -rf duk_*_kmer_tmp

#Para convertir de fastq a fasta
seqret -sequence duk_source_dir/salida_duk.fastq -outseq duk_source_dir/salida_duk.fasta

#Formateo del archivo fasta para la NN
sed 's/ /_/g' duk_source_dir/salida_duk.fasta | sed 's/|/+/g' > sequences.fa.tmp
rm -f class_nn_ctxm.txt
for sec in `grep '>' sequences.fa.tmp | sed 's/>//g'`
do
     seqret -sequence sequences.fa.tmp:${sec} -outseq sec.fa.tmp > /dev/null 2>&1
     grupo=`grep '>' sec.fa.tmp | cut -d "+" -f2 | cut -d "-" -f2`
     sec2=`grep -v '>' sec.fa.tmp | sed ':a;N;$!ba;s/\n/ /g'`
     echo "${sec2},${grupo}" >> training_nn_ctxm.txt
     rm -f sec.fa.tmp
done
rm -f *.tmp
#ejecución de script en python con la NN implementada en TensorFlow.



------------------------------------ Code to the binarization of DB CTX-M reference
# importamos la librerias

import time
import numpy as np
import pandas as pd
# Funcion para vectorizar la secuencia.

def vectorizeSequence(seq):
   # the order of the letters is not arbitrary.
   # Flip the matrix up-down and left-right for reverse compliment
   ltrdict = {'A': '1000', 'C': '0100', 'G': '0010', 'T': '0001','N': '0000'}
   return [ltrdict[x] for x in seq]

origen_training_ctxm = open("training_nn_ctxm.txt", "r")#abrimos el archivo de
entrenamiento
lines = origen_training_ctxm.readlines() #Leemos cada linea del archivo.
#salida_training_ctxm = open("sequences_nn_prueba.txt", "w") #escribimos el archivo de
salida para el entrenamiento.
long_max = max([len(x.split(',')[0]) for x in lines] )

#print(long_max)

#Recorremos cada linea para dar formato al archivo de salida o de entranemiento
y_data = []
X_data = []
for line in lines:
```

```
    seq = line.split(",")[0]
    if (len(seq) < long_max):
        dif = long_max-len(seq)
        N = ''.join(['N' for x in range(dif)])
        seq += N

    out = float(line.split(",")[1])
    y_data.append(out)
    grupo_ctxm = int (line.split(",")[1])/100

    salida = vectorizeSequence(seq)

    vector = []
    for k in range(len(salida)):
        if salida[k] == '1000':
           vector.append(1)
           vector.append(0)
           vector.append(0)
           vector.append(0)
        if salida[k] == '0100':
            vector.append(0)
            vector.append(1)
            vector.append(0)
            vector.append(0)
        if salida[k] == '0010':
            vector.append(0)
            vector.append(0)
            vector.append(1)
            vector.append(0)
        if salida[k] == '0001':
            vector.append(0)
            vector.append(0)
            vector.append(0)
            vector.append(1)
        if salida[k] == '0000':
            vector.append(0)
            vector.append(0)
            vector.append(0)
            vector.append(0)

    X_data.append(vector)
    #salida_training_ctxm.write(''.join(salida) + "," + str(grupo_ctxm)+'\n')

X_data = np.array(X_data)
y_data = np.array(y_data)
y_data = np.expand_dims(y_data, axis=1)

print(X_data.shape)
print(y_data.shape)
```

```
colnames = []
for i in range(X_data.shape[1]):
    namelcol = str('nc')+str(i)
    colnames.append(namelcol)
colnames.append('y')

data = np.concatenate((X_data, y_data), axis=1)
print(data.shape)

df = pd.DataFrame(data=data, columns=colnames)
print(df.head())
file_name = 'dataGen.csv'
#Cerramos los archivos abiertos y escribimos los tiempos de  procesamiento.
origen_training_ctxm.close()
df.to_csv(file_name, sep=',', encoding='utf-8')
print("Preparación del dataset lista.")
```

Point 2: Authors used the data from database but failed again to provide complete details.

Response 2: From there, and taking into account the review of the state of the art, we started from the CTX-M database already filtered by means of phylogenetic tree.

Subsequently, the reference metagenome to be studied was selected by means of a revision in the EBI-Metagenomics database (https://www.ebi.ac.uk/metagenomics/) considering the high probability that the CTX-M gene was present by reviewing four metagenomes related below which are inputs to make the prototype, only one was selected for the study:

1. https://www.ebi.ac.uk/metagenomics/projects/ERP001506

2. https://www.ebi.ac.uk/metagenomics/projects/ERP020191

3. https://www.ebi.ac.uk/metagenomics/projects/ERP016968

4. https://www.ebi.ac.uk/metagenomics/projects/ERP009131

The metagenome selected because of the high probability of finding there CTX-M for the proposed neural network training was:

- Antibiotic resistance within the preterm infant gut.
https://www.ebi.ac.uk/ena/data/view/PRJEB15257

Point 3: Authors mentioned about activation functions, “Tang” and “ELU” and I think these are “Tanh” and “RELU” respectively if not authors should define their functions.

Response 3: The required adjustments will be made

Point 4: Authors presented the ROC curves for training data and not sure whether they tested the model on independent test data.

Response 4: The required adjustments will be made

Point 5: Table 1 shows the precision test value as 1000 for ELU activation function it can't be true as this value should be in the range of 0-1.

Response 5: The required adjustments will be made

Point 6: Figure 6, legend shows data for 10 classes but the plot shows only 4 curves, what happened to the remaining 6 classes?

Response 6: The other six classes are skewed to the axes in the figure, only the 4 classes you can see touch the area under the curve.

Point 7: The language of the manuscript is in bad state and should be edited by native English speaker or professional editing service. A lot of formatting errors in the text are present throughout the manuscript, so should be fixed.

Response 7: The required adjustments will be made