Q1. Assume there are 120 students in our B565 class, and each person has a 2% of chance of carrying coronavirus. We also know that the Omicron variant dominates and let's assume that it accounts for 90% of the new Covid cases. What's the probability that the entire class is free of coronavirus and Omicron, respectively? Show your work. [10 pts]

i. The probability that the entire class is free of coronavirus:
If each student has a 2% chance of carrying coronavirus, then each student has a 98% chance of not carrying coronavirus.
Hence, if a class of 120 students doesn't have coronavirus, the probability of that happening is **0.98^120=0.08853787272** which is 8.8% approximately. (We multiply the probability of each student not having coronavirus).

ii. The probability that the entire class is free of Omicron:
If each student has a 2% chance of carrying coronavirus, then each student has a 0.02*0.9=0.018% chance of carrying Omicron variant. The probability of a student not having Omicron is 1-0.018=0.982.
Hence, if a class of 120 students doesn't have coronavirus, the probability of that happening is **0.982^120=0.0.11307810828** which is 11.3% approximately. (We multiply the probability of each student not having Omicron).

Q2. Let Ω be the space of possible outcomes of a fair die (with six sides) thrown twice. What is Ω? Let A be the event that a 4 is observed on either individual throw or the sum of both throws is at least 5. Let B be the event that the difference between the two throws is exactly two. Are A and B independent? What is the probability of B given A? What is the probability of A given B? Show your work. [15 pts]

i. Ω={(1, 1),(1, 2),(1, 3),(1, 4),(1, 5),(1, 6),(2, 1),(2, 2),(2, 3),(2, 4),(2, 5),(2, 6),(3, 1),(3, 2),(3, 3),(3, 4),(3, 5),(3, 6),(4, 1),(4, 2),(4, 3),(4, 4),(4, 5),(4, 6),(5, 1),(5, 2),(5, 3),(5, 4),(5, 5),(5, 6),(6, 1),(6, 2),(6, 3),(6, 4),(6, 5),(6, 6)}. Size of Ω is 36.

ii. A={(1, 4),(1, 5),(1, 6),(2, 3),(2, 4),(2, 5),(2, 6),(3, 2),(3, 3),(3, 4),(3, 5),(3, 6),(4, 1),(4, 2), (4, 3),(4, 4),(4, 5),(4, 6),(5, 1),(5, 2),(5, 3),(5, 4),(5, 5),(5, 6),(6, 1),(6, 2),(6, 3),(6, 4),(6, 5),(6, 6)}
n(A)=36.

B={(1,3),(2,4),(3,5),(4,6),(3,1),(4,2),(5,3),(6,4)}.
n(B)=8

Since, A∩B is not an empty set, A and B are not independent.
Set A∩B={(2,4),(3,5),(4,6),(4,2),(5,3),(6,4)}. n(A∩B)=6.

P(A)=n(A)/n(Ω)=30/36=5/6

P(B)=n(B)/n(Ω)=30/36=8/36=2/9

P(A∩B)= n(A∩B)/n(Ω)=6/36=1/6

iii. **P(B|A)=(P(A|B)P(B))/P(A)=P(A∩B)/P(A)=(1/6)/(5/6)=1/5**

iv. **P(A|B)=(P(B|A)P(A))/P(B)=P(A∩B)/P(B)=(1/6)/(2/9)=1/27**

Q3. Recall that given two vectors x and y of n dimensions, their cosine similarity, Euclidean distance, and correlation can be computed as follows. [15 pts]
Cosine similarity, cos(x, y) = (x · y)/||x||.||y||
Euclidean distance, d(x, y) = (∑ n i=1(xi − yi)^2)^1/2
Correlation, corr(x, y) = Sxy/(SxSy)
=(∑ n i=1 (xi − $\bar{x}$)(yi − $\bar{y}$))/ (∑ n i=1 (xi − $\bar{x}$)^2)^1/2 .(∑ n i=1 (yi − $\bar{y}$) ^2)^1/2

Sets can also be represented as vectors of zeros and ones, so for those vectors, Jaccard similarity (intersection over union) can be used. For the following vectors x and y, calculate the indicated similarity or distance measures. Show the steps.
• x = (1, 1, 1, 1), y = (2, 2, 2, 2) cosine, correlation and Euclidean.
• x = (0, 1, 0, 1), y = (1, 0, 1, 0), cosine, correlation, Euclidean, Jaccard
• x = (0, −1, 0, 1), y = (1, 0, −1, 0), cosine, correlation, Euclidean
• x = (1, 1, 0, 1, 0, 1), y = (1, 1, 1, 0, 0, 1) cosine, correlation, Jaccard
• x = (2, −1, 0, 2, 0, −3), y = (−1, 1, −1, 0, 0, −1) cosine, correlation

i. x = (1, 1, 1, 1), y = (2, 2, 2, 2)

a. Cosine Similarity:
   x.y=1.2+1.2+1.2+1.2=8
   ||x||=(1^2+1^2+1^2+1^2)^(1/2)=2
   ||y||=(2^2+2^2+2^2+2^2)^(1/2)=4
   cos(x, y) = (x · y)/||x||.||y||=8/(2*4)=1

b. Correlation:
   $\bar{x}$=(∑ n i=1(xi))/n=(1+1+1+1)/4=1
   $\bar{y}$=(∑ n i=1(yi))/n=(2+2+2+2)/4=2
   (∑ n i=1 (xi − $\bar{x}$)(yi − $\bar{y}$))=(1-1)*(2-2)+(1-1)*(2-2)+(1-1)*(2-2)+(1-1)*(2-2)=0
   (∑ n i=1 (xi − $\bar{x}$)^2)=(1-1)^2+(1-1)^2+(1-1)^2+(1-1)^2=0
   (∑ n i=1 (xi − $\bar{x}$)^2)^1/2=0
   (∑ n i=1 (yi − $\bar{y}$)^2)=(2-2)^2+(2-2)^2+(2-2)^2+(2-2)^2=0
   (∑ n i=1 (yi − $\bar{y}$)^2)^1/2=0
   Since, both the numerator and the denominator are not defined, the correlation for vectors x and y are not defined.

c. Euclidean distance:
   ∑ n i=1(xi − yi)^2=(1-2)^2+(1-2)^2+(1-2)^2+(1-2)^2=4*(-1)^2=4
   (∑ n i=1(xi − yi)^2)^1/2 =4^(1/2)=2

ii. x = (0, 1, 0, 1), y = (1, 0, 1, 0)

a. Cosine Similarity:

x.y=0.1+1.0+0.1+1.0=0

||x||=(0^2+1^2+0^2+1^2)^(1/2)=2^(1/2)

||y||=(1^2+0^2+1^2+0^2)^(1/2)=2^(1/2)

cos(x, y) = (x · y)/||x||.||y||=0

b. Correlation:

$\bar{x}$=(∑ n i=1(xi))/n=(0+1+0+1)/4=1/2

$\bar{y}$=(∑ n i=1(yi))/n=(1+0+1+0)/4=½

(∑ n i=1 (xi − $\bar{x}$)(yi − $\bar{y}$))

=(0-1/2)(1-1/2)+(1-1/2)(0-1/2)+(0-1/2)(1-1/2)+(1-1/2)(0-1/2)=(-1/4)*4=-1

(∑ n i=1 (xi − $\bar{x}$)^2)=(0-1/2)^2+(1-1/2)^2+(0-1/2)^2+(1-1/2)^2=4*1/4=1

(∑ n i=1 (xi − $\bar{x}$)^2)^1/2=1

(∑ n i=1 (yi − $\bar{y}$)^2)=(1-1/2)^2+(0-1/2)^2+(1-1/2)^2+(0-1/2)^2=4*1/4=1

(∑ n i=1 (yi − $\bar{y}$)^2)^1/2=1

Hence,

corr(x, y) = Sxy/(SxSy)= -1/1*1=-1

c. Euclidean distance:

∑ n i=1(xi − yi)^2=(0-1)^2+(1-0)^2+(0-1)^2+(1-0)^2=4

(∑ n i=1(xi − yi)^2)^1/2 =4^(1/2)=2

d. Jaccard Coefficient:

JC= (f11)/(f01+f10+f11)=0/(2+2+0)=0

iii. x = (0, −1, 0, 1), y = (1, 0, −1, 0)

a. Cosine Similarity:

x.y=0.1+-1.0+0.-1+1.0=0

||x||=(0^2+(-1)^2+0^2+1^2)^(1/2)=2^(1/2)

||y||=(1^2+0^2+(-1)^2+0^2)^(1/2)=2^(1/2)

cos(x, y) = (x · y)/||x||.||y||=0

b. Correlation:

$\bar{x}$=(∑ n i=1(xi))/n=(0+-1+0+1)/4=0

$\bar{y}$=(∑ n i=1(yi))/n=(1+0+-1+0)/4=0

(∑ n i=1 (xi − $\bar{x}$)(yi − $\bar{y}$))

=(0)(1)+(-1)(0)+(0)(-1)+(1)(0)=0

(∑ n i=1 (xi − $\bar{x}$)^2)=(0)^2+(-1)^2+(0)^2+(1)^2=2

$(\sum_{i=1}^{n} (x_i - \bar{x})^2)^{1/2} = 2^{1/2}$
$(\sum_{i=1}^{n} (y_i - \bar{y})^2) = (1)^2+(0)^2+(-1)^2+(0)^2 = 2$
$(\sum_{i=1}^{n} (y_i - \bar{y})^2)^{1/2} = 2^{1/2}$

Hence,
corr(x, y) = Sxy/(SxSy)= 0

c. Euclidean distance:
$\sum_{i=1}^{n}(x_i - y_i)^2 = (0-1)^2+(-1-0)^2+(0-(-1))^2+(1-0)^2 = 4$
$(\sum_{i=1}^{n}(x_i - y_i)^2)^{1/2} = 4^{(1/2)} = 2$

iv. x = (1, 1, 0, 1, 0, 1), y = (1, 1, 1, 0, 0, 1)

a. Cosine Similarity:
$x.y = 1.1+1.1+0.1+1.0+0.0+1.1 = 3$
$||x|| = (1^2+1^2+0^2+1^2+0^2+1^2)^{(1/2)} = 4^{(1/2)} = 2$
$||y|| = (1^2+1^2+1^2+0^2+0^2+1^2)^{(1/2)} = 4^{(1/2)} = 2$
$\cos(x, y) = (x \cdot y)/||x||.||y|| = 3/(2*2) = 0.75$

b. Correlation:
$\bar{x} = (\sum_{i=1}^{n}(x_i))/n = (1+1+0+1+0+1)/6 = 4/6 = 2/3$
$\bar{y} = (\sum_{i=1}^{n}(y_i))/n = (1+1+1+0+0+1)/6 = 4/6 = 2/3$

$(\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}))$
$= (1-2/3)(1-2/3)+(1-2/3)(1-2/3)+(0-2/3)(1-2/3)+(1-2/3)(0-2/3)+(0-2/3)(0-2/3)+(1-2/3)(1-2/3)$
$= 1/9+1/9-2/9-2/9+4/9+1/9 = 3/9 = 1/3$

$(\sum_{i=1}^{n} (x_i - \bar{x})^2) = (1-2/3)^2+(1-2/3)^2+(0-2/3)^2+(1-2/3)^2+(0-2/3)^2+(1-2/3)^2 = 1/9+1/9+4/9+1/9+4/9+1/9 = 4/3$
$(\sum_{i=1}^{n} (x_i - \bar{x})^2)^{1/2} = (4/3)^{1/2}$
$(\sum_{i=1}^{n} (y_i - \bar{y})^2) = (1-2/3)^2+(1-2/3)^2+(1-2/3)^2+(0-1/2)^2+(0-1/2)^2+(1-2/3)^2 = 1/9+1/9+1/9+4/9+4/9+1/9 = 4/3$
$(\sum_{i=1}^{n} (y_i - \bar{y})^2)^{1/2} = (4/3)^{1/2}$
Hence,
corr(x, y) = Sxy/(SxSy)= 1/3/(4/3)^1/2*(4/3)^1/2=0.25

c. Jaccard Coefficient:
JC= (f11)/(f01+f10+f11)=3/(1+1+3)=0.6

v. x = (2, −1, 0, 2, 0, −3), y = (−1, 1, −1, 0, 0, −1)

a. Cosine Similarity:

x.y=2.-1+-1.1+0.-1+2.0+0.0+-3.-1=-2-1+3=0
||x||=(2^2+-1^2+0^2+2^2+0^2+-3^2)^(1/2)=18^(1/2)
||y||=(-1^2+1^2+-1^2+0^2+0^2+-1^2)^(1/2)=4^(1/2)=2
cos(x, y) = (x · y)/||x||.||y||=0/(2*(18)^1/2)=0.0

b. Correlation:

$\bar{x}$=(∑ n i=1(xi))/n=(2-1+0+2+0-3)/6=0
$\bar{y}$=(∑ n i=1(yi))/n=(-1+1-1+0+0-1)/6=-2/6

(∑ n i=1 (xi − $\bar{x}$)(yi − $\bar{y}$))
=(2)(-1+2/6)+(-1)(1+2/6)+(0)(1+2/6)+(2)(-2/6)+(0)(+2/6)+(-3)(-1+2/6)=0

(∑ n i=1 (xi − $\bar{x}$)^2)=(2)^2+(-1)^2+(0)^2+(2)^2+0^2+(-3)^2=18
(∑ n i=1 (xi − $\bar{x}$)^2)^1/2=18^1/2
(∑ n i=1 (yi − $\bar{y}$)^2)=(-1+2/6)^2+(1+2/6)^2+(1+2/6)^2+(-2/6)^2+(+2/6)^2+(-1+2/6)^2
=10/3
(∑ n i=1 (yi − $\bar{y}$)^2)^1/2=(10/3)^1/2
Hence,
corr(x, y) = Sxy/(SxSy)= 0/((18^½)*((10/3)^1/2))=0

Q4. Analyze a microbiome dataset. [30 pts]

• The dataset is available here.

• This dataset includes the microbiome profiles of 344 people, some with type 2 diabetes, and others without. The microbiome profile for a person stores the relative abundance of different bacterial specie found in the stool sample collected from that person. The last column shows the class (with diabetes or not), and the other columns are for the relative abundances.

• Perform PCA and t-SNE on the dataset and visualize the data in 2D space. In the plots, each data point is a user.

• Report what you learn from the PCA analyses. How much variability of the data is captured by using only two dimensions? Is PCA a good approach for dimensionality reduction for this dataset? Do you see clusters of people according to their disease status?

• Does t-SNE result in a good dimensionality reduction of this dataset? Why or why not

First, we will perform PCA on this dataset

In [2]:
```
import pandas as pd
import numpy as np
microbiome = pd.read_csv("T2D_abundance.csv",delim_whitespace=True)
microbiome.head()
```

Out[2]:

	k__Archaea|p__Euryarchaeota|c__Methanobacteria|o__Methanobacteriales|f__Methanobacteria
con-001	
con-002	
con-003	
con-004	
con-005	

5 rows × 573 columns

In [3]:
```
microbiome.shape
```

Out[3]: (344, 573)

In [4]:
```
microbiome.describe()
```

Out[4]:

	k__Archaea|p__Euryarchaeota|c__Methanobacteria|o__Methanobacteriales|f__Methanobacter
count	
mean	
std	
min	
25%	
50%	
75%	
max	

8 rows × 572 columns

In [5]:
```
class_column = microbiome['Class'].tolist()
class_column
```

```
Out[5]: ['n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
```

'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',

```
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
```

```
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
               't2d',
```

't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
'n',
'n',
'n',

```
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n']
```

In [6]:
```
microbiome = microbiome.drop('Class', 1)
```

Using PCA, the data matrix is projected to its first two principal components. The projected values of the original image data are stored in a pandas DataFrame object named projected.

In [8]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(microbiome)
#the results will show that PC1 dominates for the original data
print("variance explained", pca.explained_variance_ratio_, "singular_values", p
ca.singular_values_)
#transform the data according to the PCA results
microbiome_transformed = pca.transform(microbiome)
```

```
variance explained [0.29922075 0.06936047] singular_values [337.41686876 162.45
270413]
```

In [9]:
```
projected = pd.DataFrame(microbiome_transformed,columns=['pc1','pc2'],index=ran
ge(1,345))
projected['Class'] =class_column

projected
```

Out[9]:

	pc1	pc2	Class
1	-2.943049	-2.248651	n
2	-7.215752	1.835266	n
3	24.051681	-4.850905	n
4	-8.684207	6.432183	n
5	2.565706	-3.969565	n
...	...	...	...
340	-7.294597	1.557039	n
341	-2.864927	18.011522	n
342	-8.629940	-7.938802	n
343	-6.603719	-2.611546	n
344	-8.060710	-9.186921	n

344 rows × 3 columns

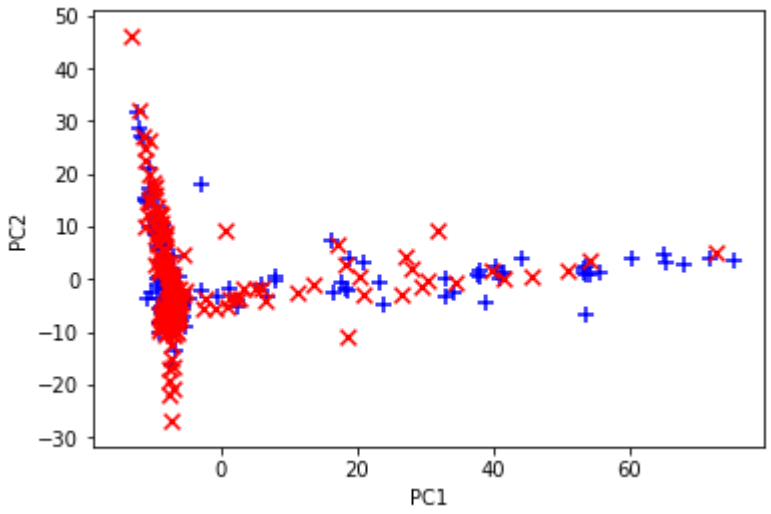
In [11]:
```
import matplotlib.pyplot as plt

colors = {'n':'b', 't2d':'r'}
markerTypes = {'n':'+', 't2d':'x'}

for classType in markerTypes:
    d = projected[projected['Class']==classType]
    plt.scatter(d['pc1'],d['pc2'],c=colors[classType],s=60,marker=markerTypes[c
lassType])

plt.xlabel("PC1")
plt.ylabel("PC2")
```

Out[11]: Text(0, 0.5, 'PC2')



Finally, we draw a scatter plot to display the projected values. We can see that the class 't2d' values overlap with the 'n' class values, in many cases. Hence, there is not much variability captured using PCA with 2 components. We can't see proper clusters of people according to the disease status. Hence, PCA is not a good approach to reduce this dataset.

Now, we will perform t-SNE on this dataset.

In [12]:
```
#t-sne:
from sklearn.manifold import TSNE

from numpy import reshape
import seaborn as sns

microbiome1 = pd.read_csv("T2D_abundance.csv",delim_whitespace=True)
microbiome1.head()
```

Out[12]:

	k__Archaea|p__Euryarchaeota|c__Methanobacteria|o__Methanobacteriales|f__Methanobacteria
con-001	
con-002	
con-003	
con-004	
con-005	

5 rows × 573 columns

In [13]:
```
class_column1 = microbiome1['Class'].tolist()
class_column1
```

```
Out[13]: ['n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
```

'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',

```
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
'n',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
```

```
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
                't2d',
```

't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
't2d',
'n',
'n',
'n',

```
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n',
 'n']
```

In [14]:
```
microbiome1 = microbiome1.drop('Class', 1)
```

In [15]:
```
x = microbiome1
y = class_column1
```

Then, we'll define the model by using the TSNE class, here the n_components parameter defines the number of target dimensions.

In [16]:
```
tsne = TSNE(n_components=2, verbose=1, random_state=123)
z = tsne.fit_transform(x)
```

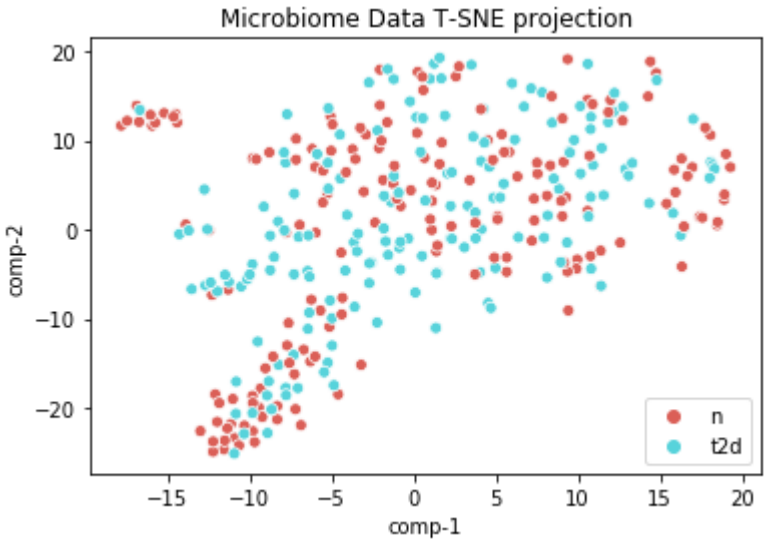
```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 344 samples in 0.137s...
[t-SNE] Computed neighbors for 344 samples in 0.504s...
[t-SNE] Computed conditional probabilities for sample 344 / 344
[t-SNE] Mean sigma: 9.567743
[t-SNE] KL divergence after 250 iterations with early exaggeration: 67.894165
[t-SNE] KL divergence after 1000 iterations: 0.865274
```

Now, we will visualize the data.

In [17]:
```
df = pd.DataFrame()
df["y"] = y
df["comp-1"] = z[:,0]
df["comp-2"] = z[:,1]

sns.scatterplot(x="comp-1", y="comp-2", hue=df.y.tolist(),palette=sns.color_pal
ette("hls", 2),data=df).set(title="Microbiome Data T-SNE projection")
```

Out[17]:
```
[Text(0.5, 1.0, 'Microbiome Data T-SNE projection')]
```



t-SNE offers slightly better separation of the classes and there is not much overlap between the data points.

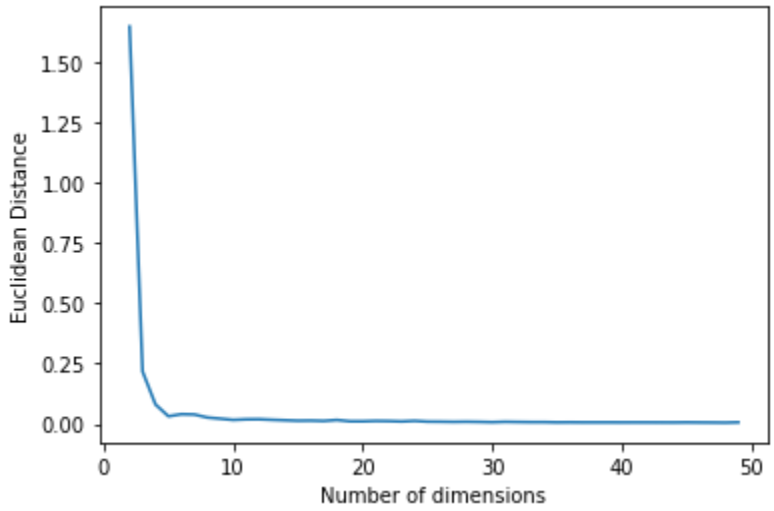
Q5. The plot below was used to demonstrate the Curse of Dimensionality. Implement a code to simulate your own data, and generate your special plot of curse of dimensionality. Try dimensions from 2 to 50 with a step size of 1. And for each dimension, randomly generate 500 data points. Use Euclidean distance. [30 pts]

We are assuming A is the set of points given and B is the reference point, if you compute the distance from B to each of the points in B, the difference between the maximum distance and the minimum distance goes to 0.

In [20]:
```
import numpy as np
import matplotlib.pyplot as plt
import os
import math

deltas = []
for N in range(2,50):
    # Generate 500 random points in N dimensions.
    A = [np.random.randint(-100, 100, N) for _ in range(500)]
    B = np.random.randint(-100,100,N)
    diffs = [np.linalg.norm(a-B) for a in A]
    mxd = max(diffs)
    mnd = min(diffs)
    delta = math.log10(mxd-mnd)/mnd
    deltas.append( delta )

plt.plot(range(2,50),deltas)
plt.xlabel('Number of dimensions')
plt.ylabel('Euclidean Distance')
plt.show()
```



In [ ]: