Equipe cancer - LIVRABLE 3

Analyse du jeu de données:

Extraction des données

Étant donné que le fichier csv présentait de nombreux trous ou problèmes de formatage, nous avons dû réorganiser nos données avant de pouvoir les utiliser.
Pour cela nous avons utilisé un programme python et power query afin de nettoyer la donnée et d'avoir directement une forme X (les textes) et y (les types de cancer).
Nous avons aussi décidé de remplacer les noms de cancer par des indices.

```python
import pandas as pd
import numpy as np

dataFrame = pd.read_csv('alldata_1_for_kaggle.csv',sep = ",", encoding='latin')
print(dataFrame)
arr = np.array(dataFrame.values)

for val in arr:
    if val[1] == 'Thyroid_Cancer':
        val[1] = 0
    elif val[1] == 'Colon_Cancer':
        val[1] = 1
    elif val[1] == 'Lung_Cancer':
        val[1] = 2

lung = 'Lung_Cancer'
for i in range(len(arr)):

   if isinstance(arr[i][0], int) == False and isinstance(arr[i][0], float) == False :
      arr[i][1] = arr[i][0][len(lung):]
      arr[i][0] = i
      arr[i][2] = 2


output = pd.DataFrame(arr)
output.to_excel('DonneesCancer.xlsx');
```

Ensuite nous avons importé le fichier excel dans colab afin d'effectuer les wordclouds et les analyses.

```
[37] import pandas as pd
     import numpy as np
     dataFrame = pd.read_excel('/content/DonneesCancer.xlsx')
     arr = np.array(dataFrame.values)
     arr = arr[1:]
```

```
[38] textLung = ''
     textColon = ''
     textThyroid = ''
     autre = ''
     for val in arr:
       if val[1] == 0:
         textThyroid += val[0]
       elif val[1] == 1:
         textColon += val[0]
       elif val[1] == 2:
         textLung += val[0]
```

Traitement des données

Lien du Google Colab : (Je vais bientot le changer car colab de test)

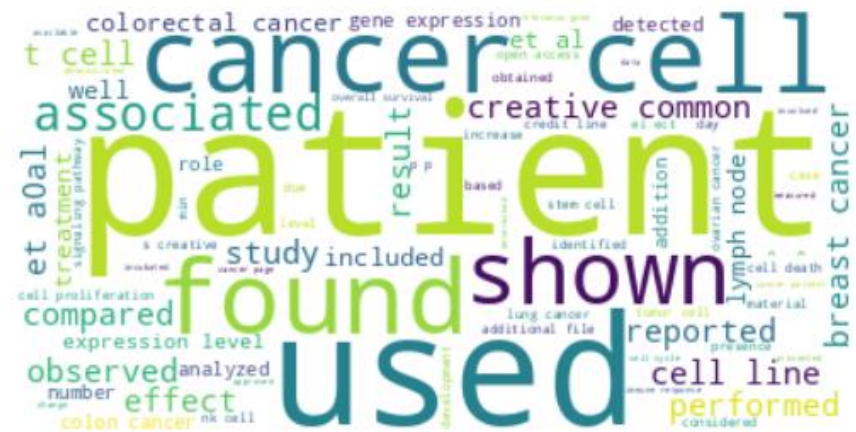
```
[ ] import matplotlib.pyplot as plt
    from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

```
[ ]
    wordcloud = WordCloud(background_color="white").generate(textColon)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

    wordcloud = WordCloud(background_color="white").generate(textLung)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

    wordcloud = WordCloud(background_color="white").generate(textThyroid)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

    # Pour lung
    # Pour thyroid
    # Pour colon
```

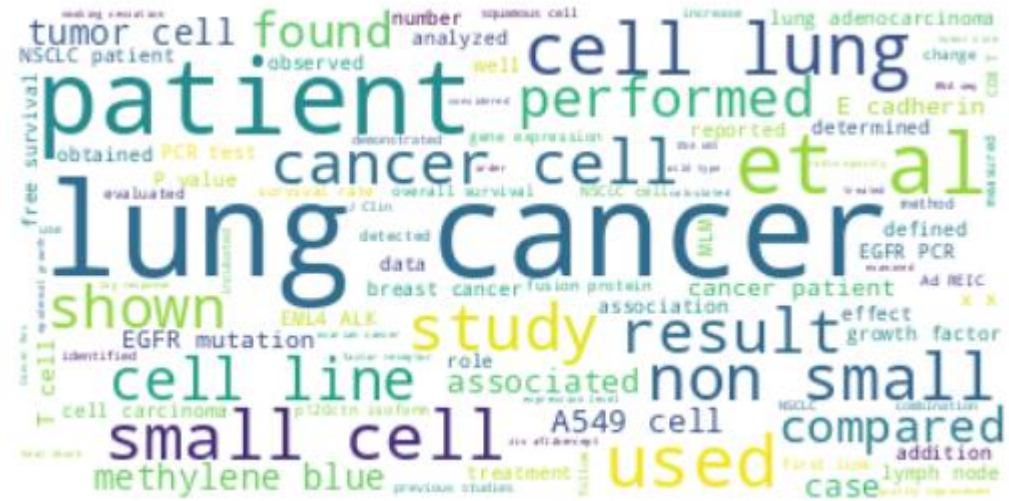
Livrable3.ipynb

Au cours de notre exploration du jeu de données nous avons utilisé la librairie des WordCloud pour identifier les termes prédominants associés aux textes. Ces visualisations nous ont permis de mettre en évidence les aspects les plus importants du jeu de données.

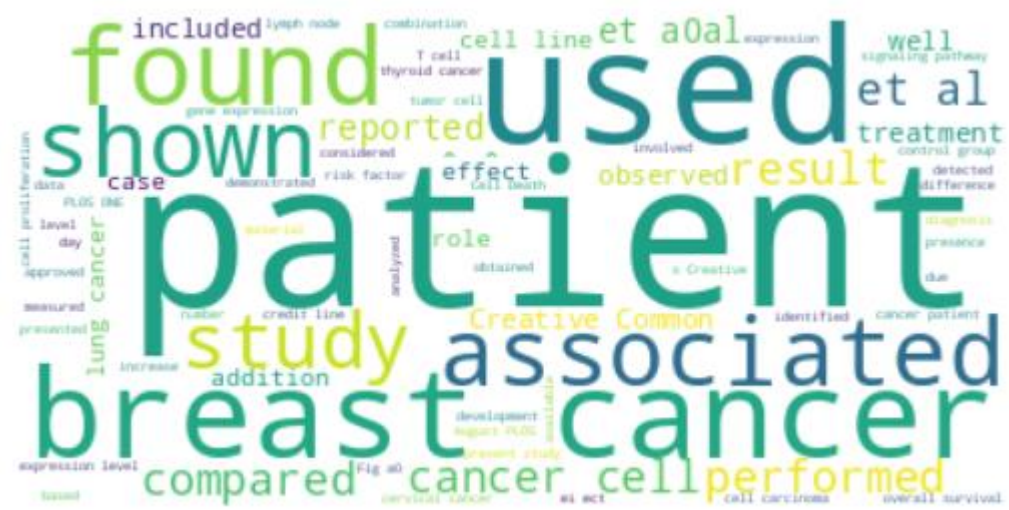
Wordcloud : Colon



Wordcloud : Lung



Wordcloud : Patient



Le Word Cloud nous a permis de mettre en évidence les éléments clés du jeu de données de manière visuelle. Cette approche a facilité l'identification des tendances, les mots plus fréquents dans le Word Cloud vont nous aider à formuler des questions pertinentes.

En plus du word cloud nous avons essayé de représenter les 10 mots les plus fréquents à l'aide d'un diagramme en bâton l'objectif était de fournir l'objectif était de fournir un aspect plus formel et statistique.
Pour cela nous avons utilisé keras et nltk (Natural Language Toolkit) afin d'avoir nos 10 mots.

```python
from collections import OrderedDict
from nltk.corpus import stopwords
import nltk
#Permet d'avoir tout les mots courants
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

#Création d'une fonction renvoyant les 10 mots les plus utilisés
def top10(dico):
  #On va d'abord filtrer pour enlever les mots courants
  filtered_word_counts = {word: count for word, count in dico.items() if word not in stop_words}
  sorted_word_counts = OrderedDict(sorted(filtered_word_counts.items(), key=lambda x: x[1], reverse=True))
  #Selectionne les 10 derniers après avoir trié le dictionnaire
  top_10_words = list(sorted_word_counts.items())[:10]
  return top_10_words
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```python
from keras.preprocessing.text import Tokenizer
token = Tokenizer()
#Transforme les textes en tableau
tabColon = textColon.split(" ")
tabLung = textLung.split(" ")
tabThyroid = textThyroid.split(" ")
#Afin de ne pas se repeter dans le code init d'un tableau avec les tableaux

tabCancers = [tabColon,tabLung,tabThyroid]
for t in tabCancers:
  token.fit_on_texts(t)
  top10tmp = top10(token.word_counts)
  print("-----------------------------------------------")
  for mot, nombre in top10tmp:
        print(f"{mot}: {nombre}")
```

Après l'analyse des 10 mots les plus fréquents nous avons remarqué que les mêmes mots revenaient pour les 3 types de cancer.

```
------------------------------------------------
cells: 56680
cancer: 54990
patients: 41720
□: 41540
cell: 39850
expression: 35240
study: 25810
fig: 24220
data: 22610
tumor: 21040
------------------------------------------------
cancer: 90923
cells: 79370
patients: 62970
cell: 58629
expression: 48420
□: 42760
0: 40100
study: 38060
tumor: 36235
lung: 35849
------------------------------------------------
cancer: 148689
cells: 124960
patients: 117923
cell: 91559
□: 90611
expression: 81630
study: 68641
p: 58641
data: 56961
tumor: 55013
```

Les résultats indiquent que les trois types de cancer (colon, poumon et thyroïde) ont des similitudes dans les termes les plus couramment utilisés pour décrire les cas. Les mots utilisés pour décrire les trois types de cancer peuvent indiquer des tendances ou des caractéristiques communes.

Pour la suite :

Nous avons également essayé de faire un fit avec quelques textes du dataset cependant nous avons vite remarqué que l'exécution ne pouvait aboutir en raison d'une limitation de la RAM. Ce qui sera à surveiller lors du fine tuning du modèle