**Code Document for *Extraction*** [***Medical Transcriptions***](https://www.kaggle.com/code/leekahwin/text-analytic-n-gram-feature-extraction/data)

**to *structured data***

Oleh:

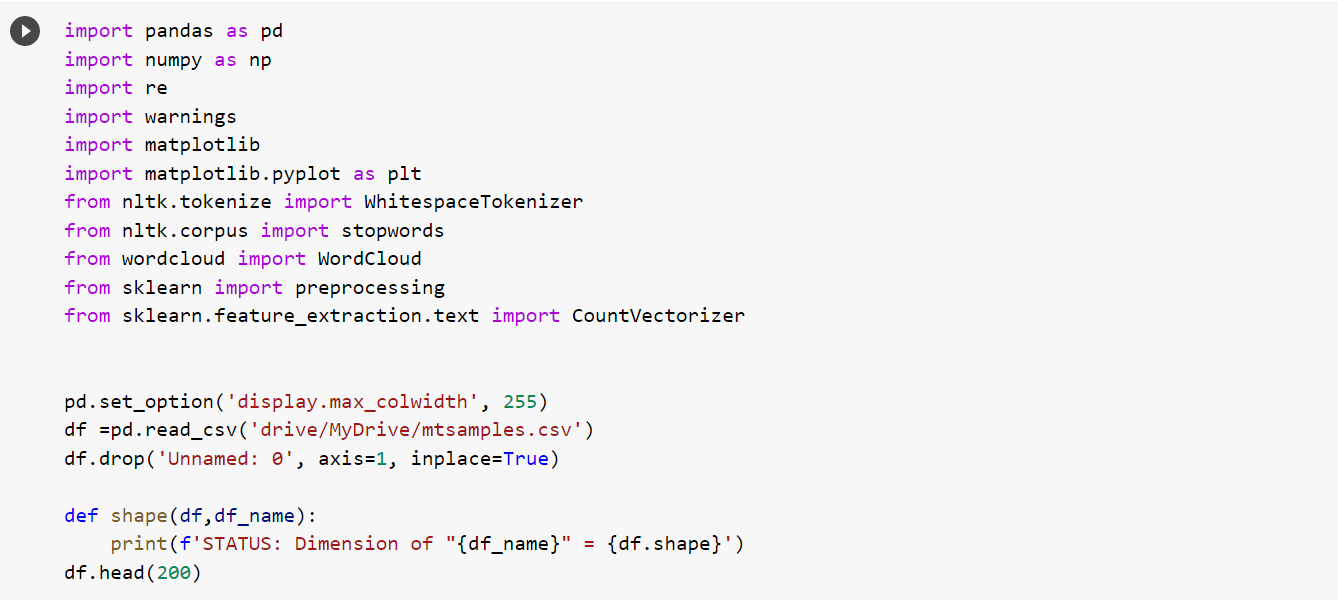
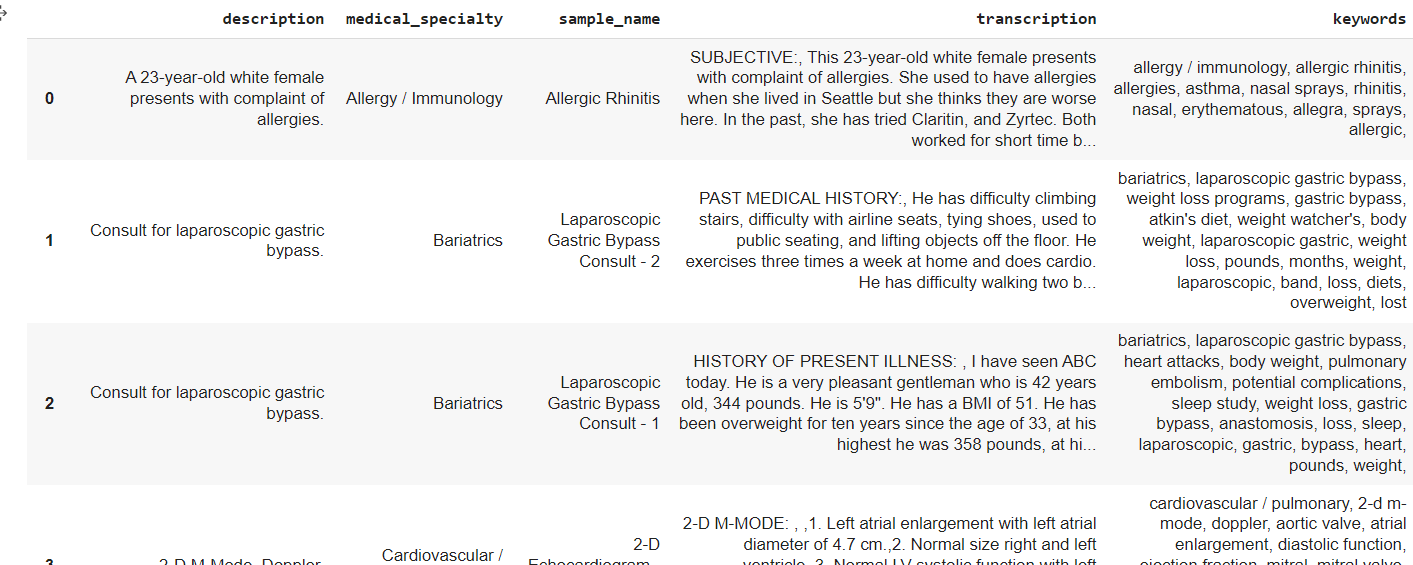
Satriyo Kristanto

22/502031/PPA/06411

1. Import Functions

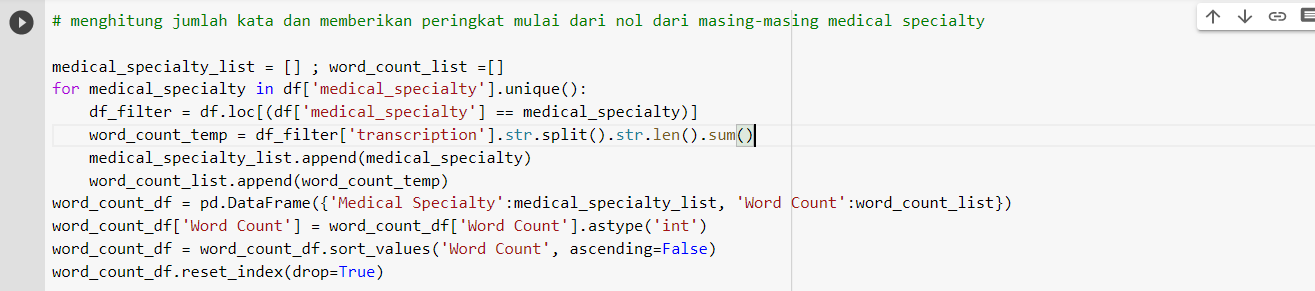
Dataset 🡪 mtsamples.csv

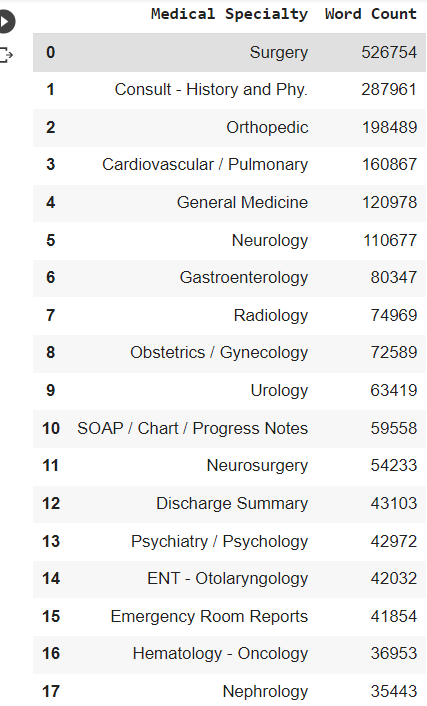
* 1. Deklarasi **Import Functions**

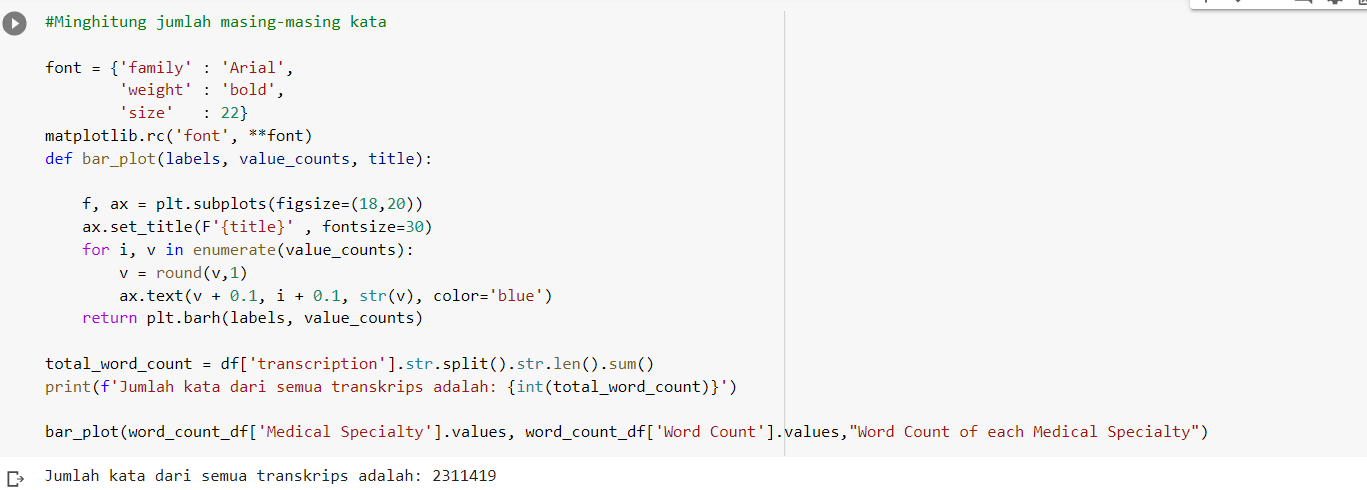
 Result of import function at least 200 row and 5 coloumn

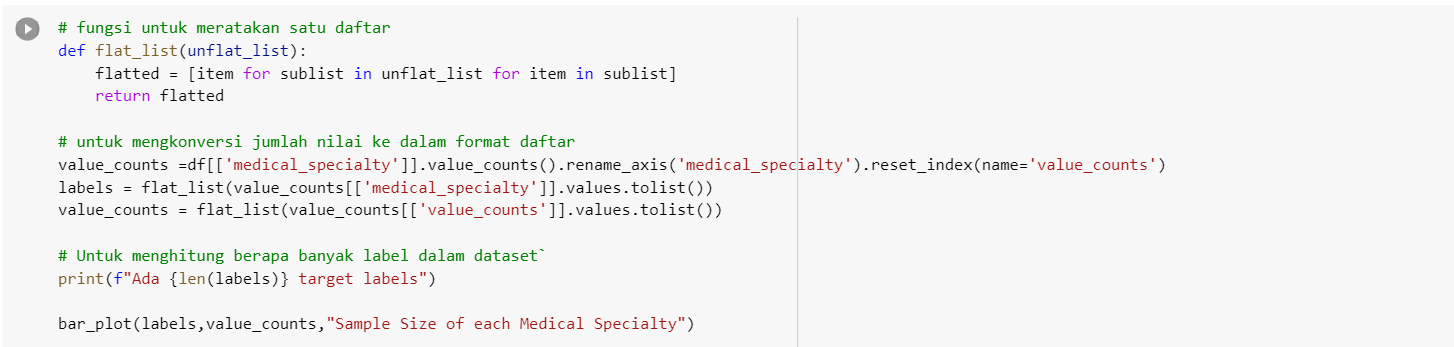
# Word Counts of Each Medical Specialty

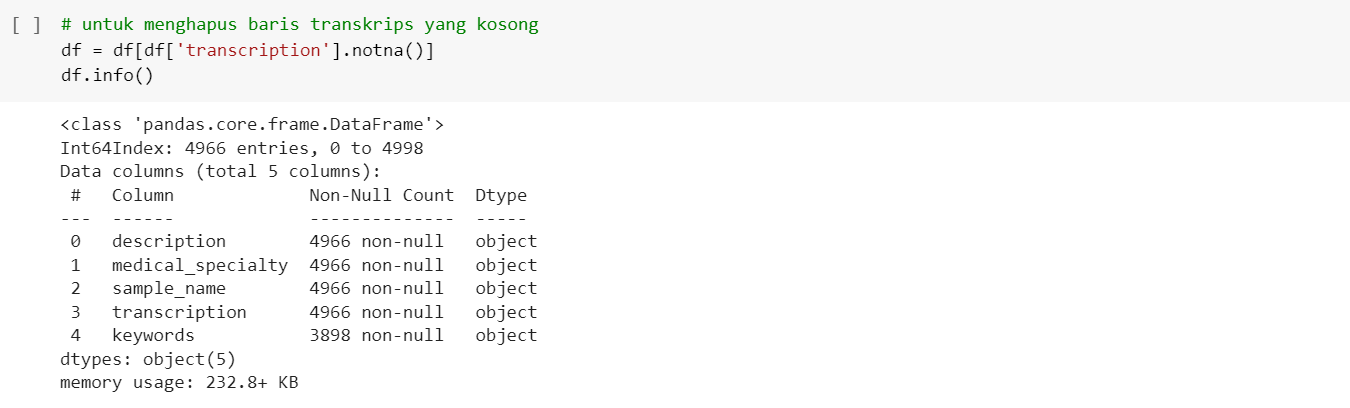
**Untuk menanyakan data, saya ingin tahu bagaimana ukuran dataset dan juga untuk memberi peringkat nilai nol dalam urutan menurun**

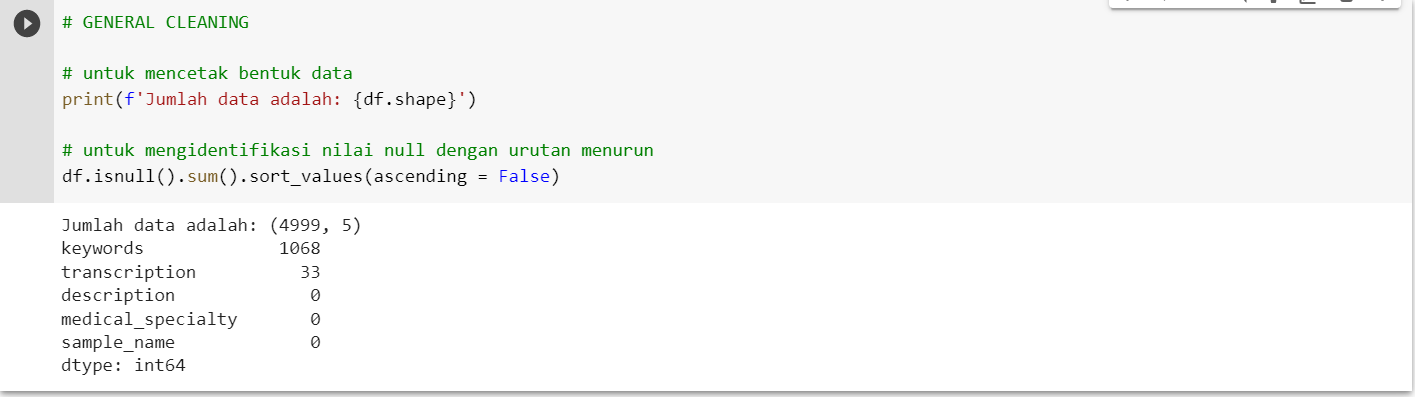
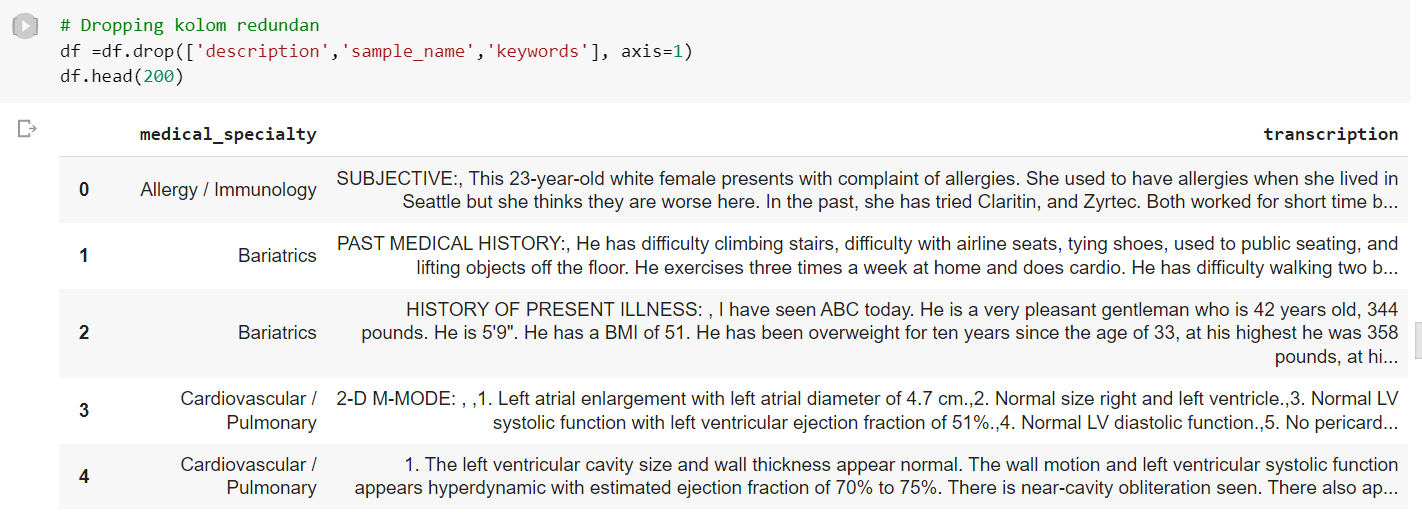






* 1. Sample Size of Each Medical Specialty
  2. General Cleaning

Satu detail penting adalah menemukan ada 2 baris yang tidak mengandung transkripsi. Mereka harus dihapus karena transkripsi adalah satu-satunya prediktor kami dalam tugas klasifikasi teks ini**.**

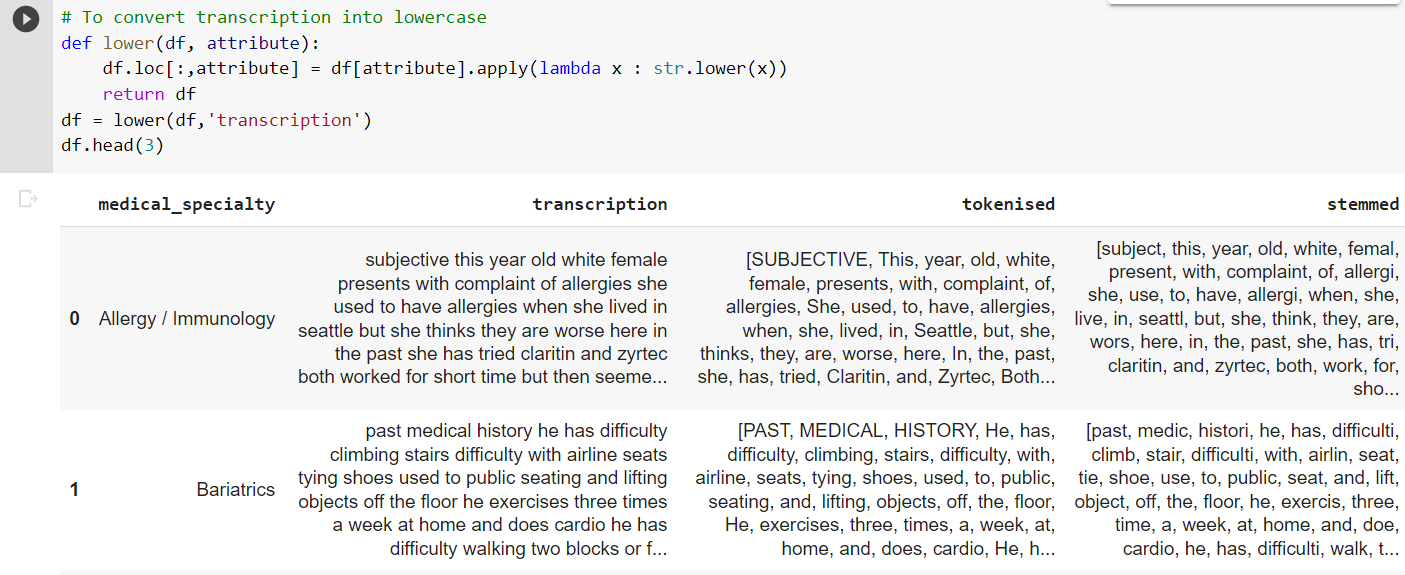
**Setelah dropping nilai null, tidak ada nilai null untuk atribut transkripsi. Sekarang telah sedikit menurun menjadi 4966 pengamatan dalam dataset ini**

The target labels (or the topic) is the 'medical\_specialty' attribute. Now, let's identify how is the value counts of the target labels, and as well visualise it in a bar chart. In order to visualise in matplotlib, function of flattening list is defined in order to put the target value counts into the matplotlib function

2.0. Text Normalisation

**Data normalisation will be conducted for the trascription. One of the reasons is to convert the transcript into standard format, which important for data extraction later. In this data normalisation task, following task will be executed, which are:**

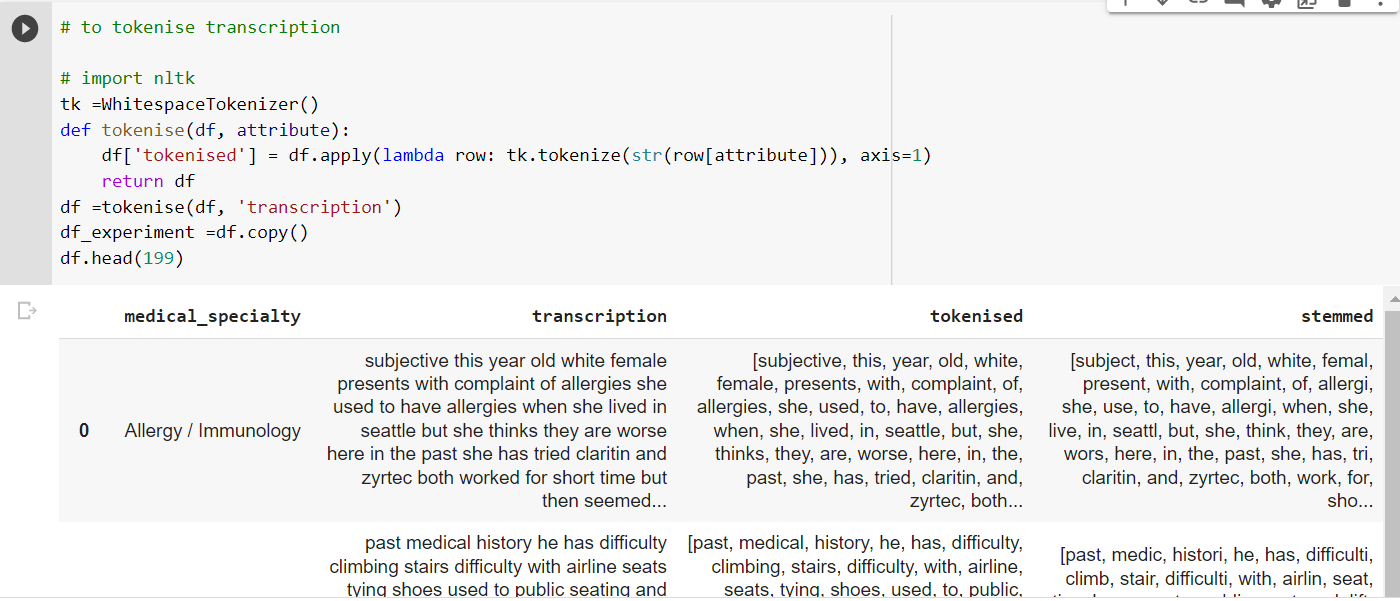
1. Lowe Case
2. Removing punctuation and numbers
3. Tokenisation of the transcription
4. Stemming
5. Remove Stop Words

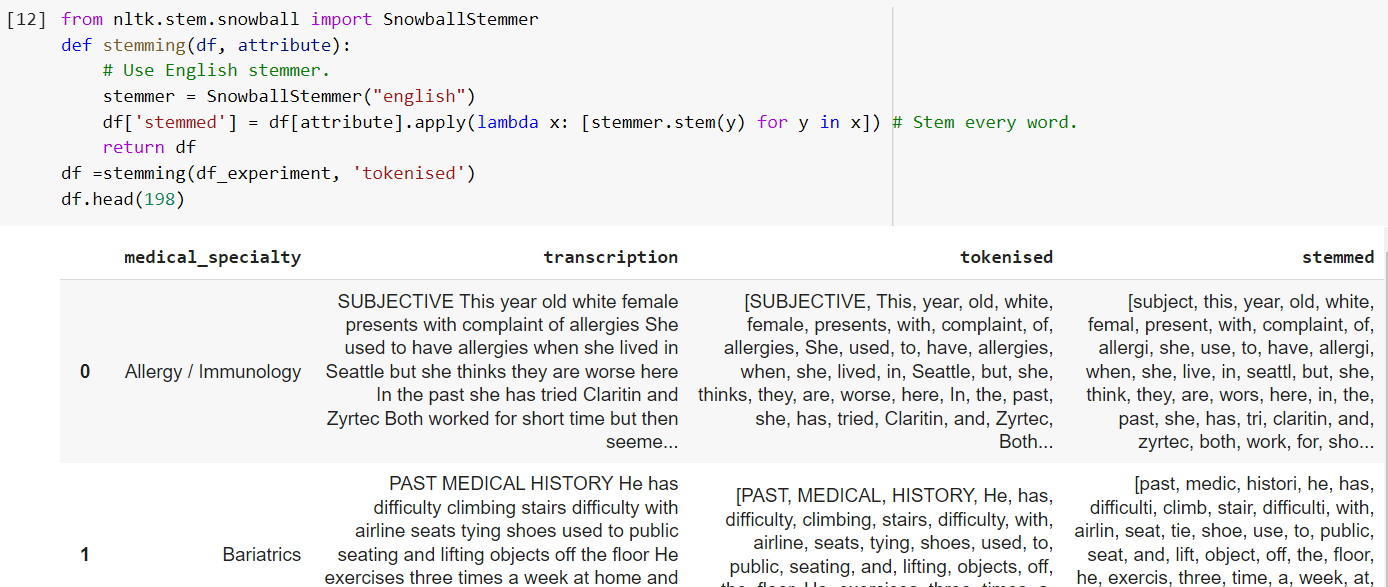
2.1. Lower Case

2.2. Removing punctuation and numbers

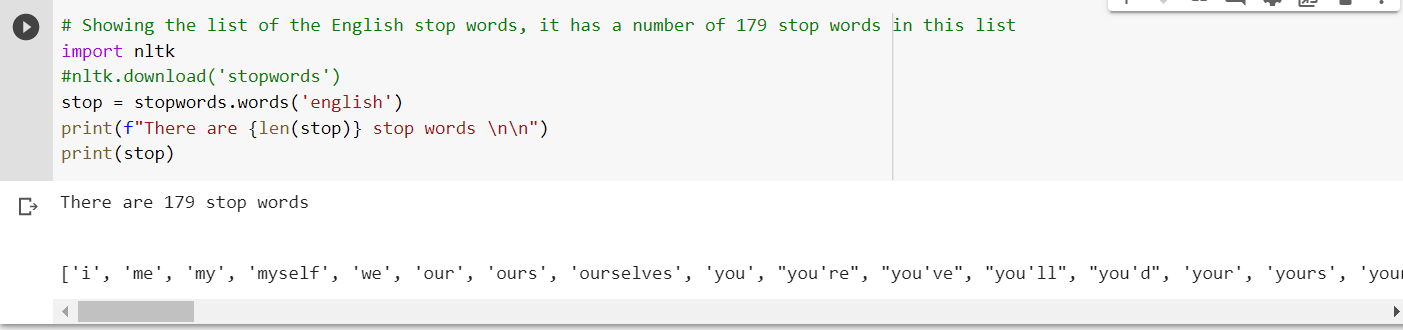
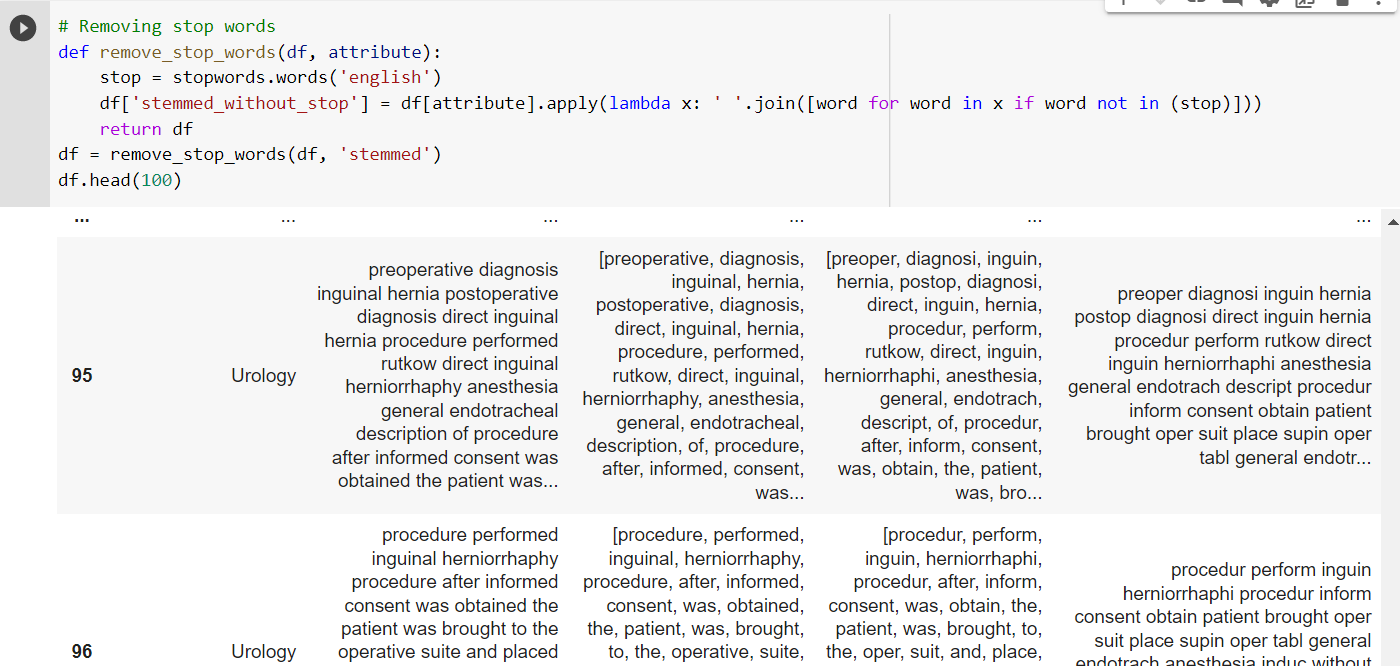


2.3. Tokenisation

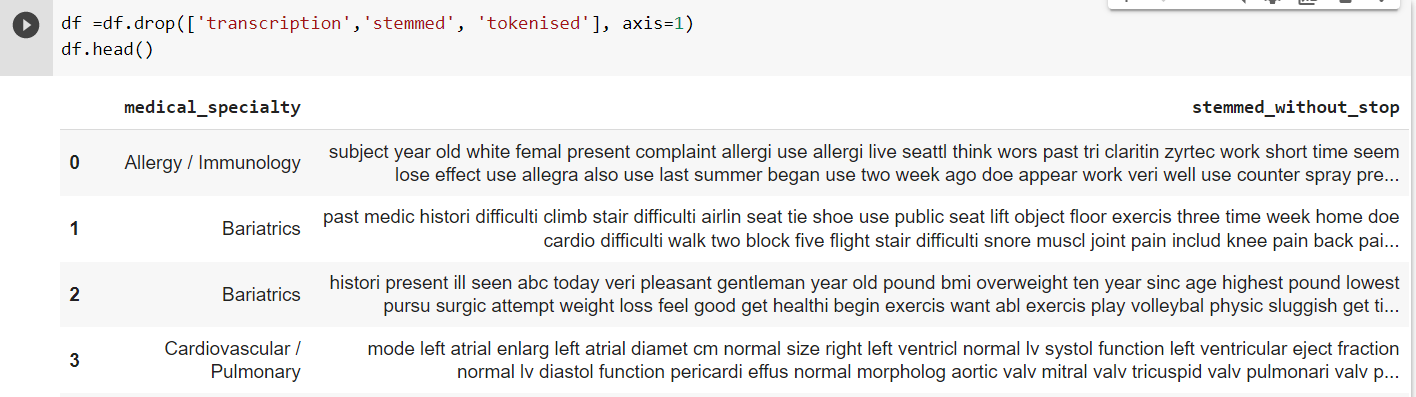


2.4. Stemming

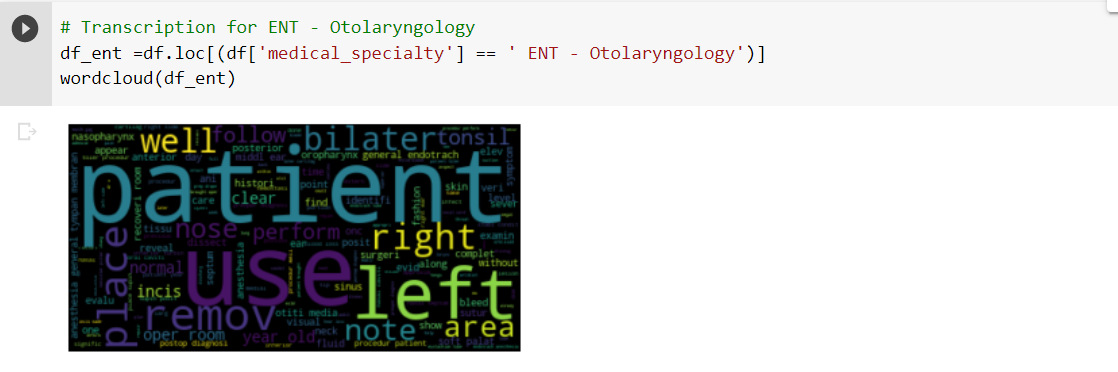
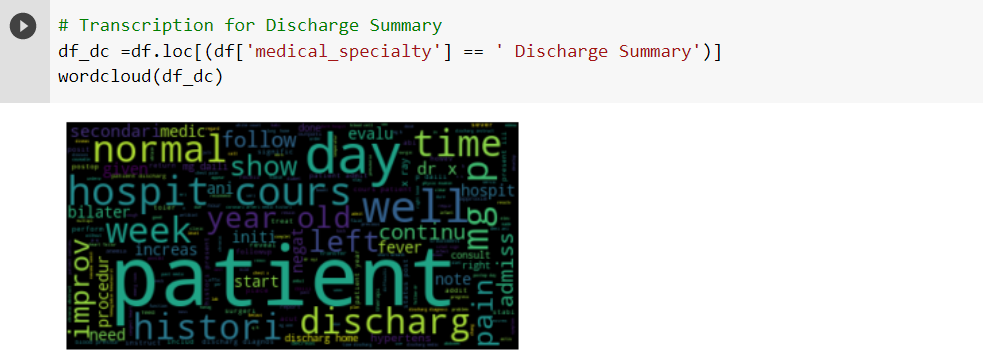
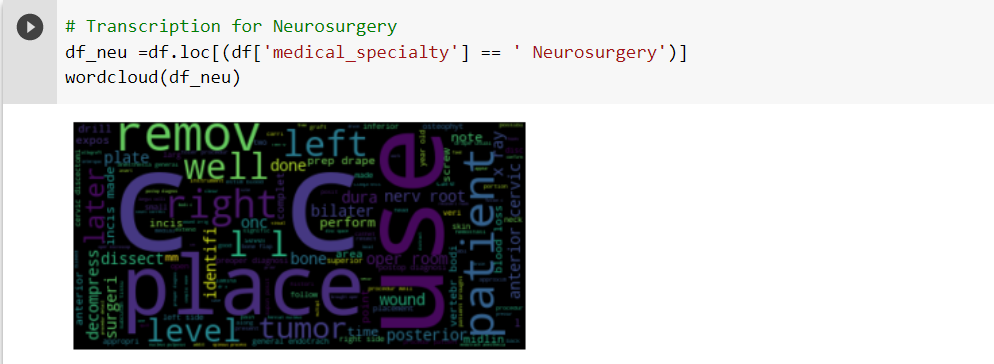
2.5. Stop Words Removal

**Removing stop words from the feature space, otherwise it will affect the classifier performance as the collection frequency is often high**

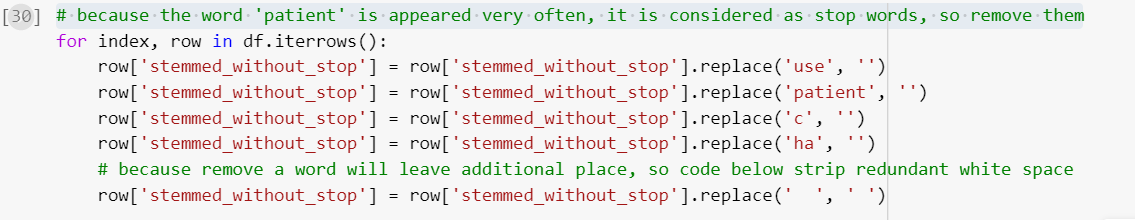
**After the 5 data normalisation steps, each transcription record is now in a standard format, which is ready for the n-gram features extraction later. Hence, we should use the attribute 'stemmed\_withou\_stop' as the predictor attribute and drop other redundant attributes, namely 'transcription', 'tokenized\_transcription' and 'stemmed'.**



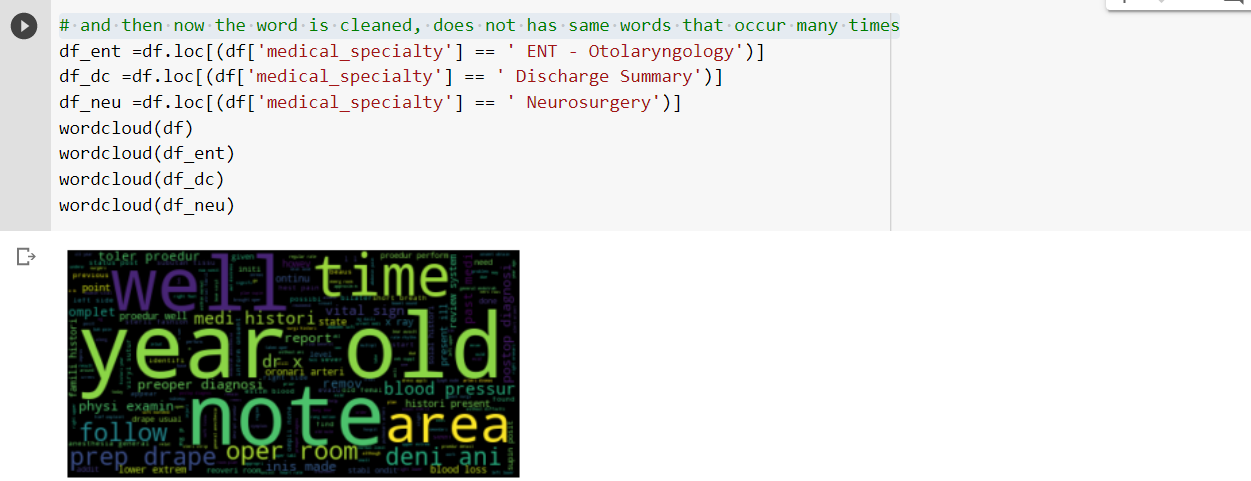
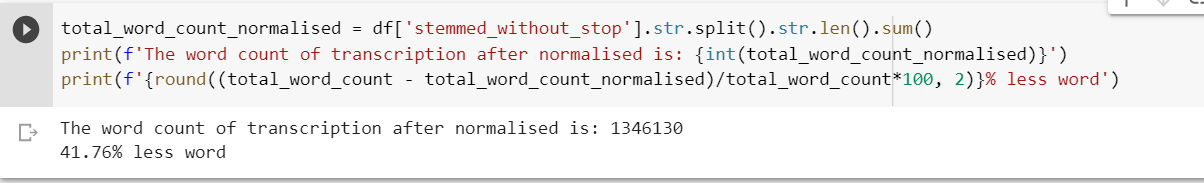
**As we can see from the dataframe above, the target column is still in text form. In order to be processed by machine, one should label encode it into numerical format. Hence sklearn preprocessing will be imported in order to label encode the target column, which is as shown below. Let show the word clouds first.**

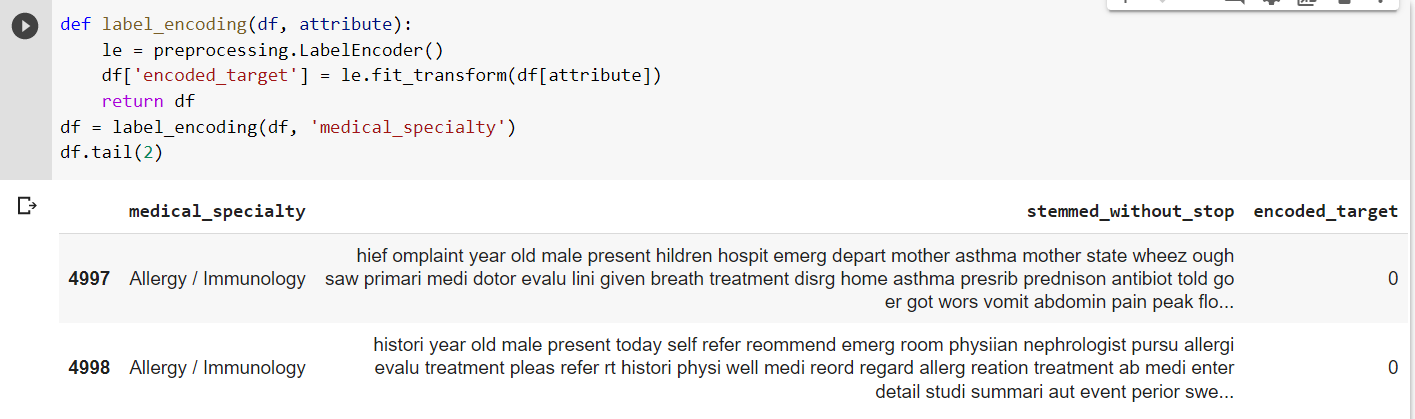
**2.6. Visualising** **Word** **Clouds**

# because the word 'patient' is appeared very often, it is considered as stop words, so remove them



# and then now the word is cleaned, does not has same words that occur many times

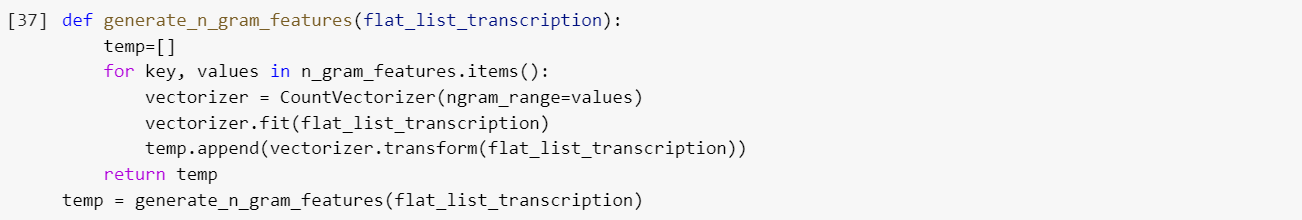
# **The word count of transcription after normalised**

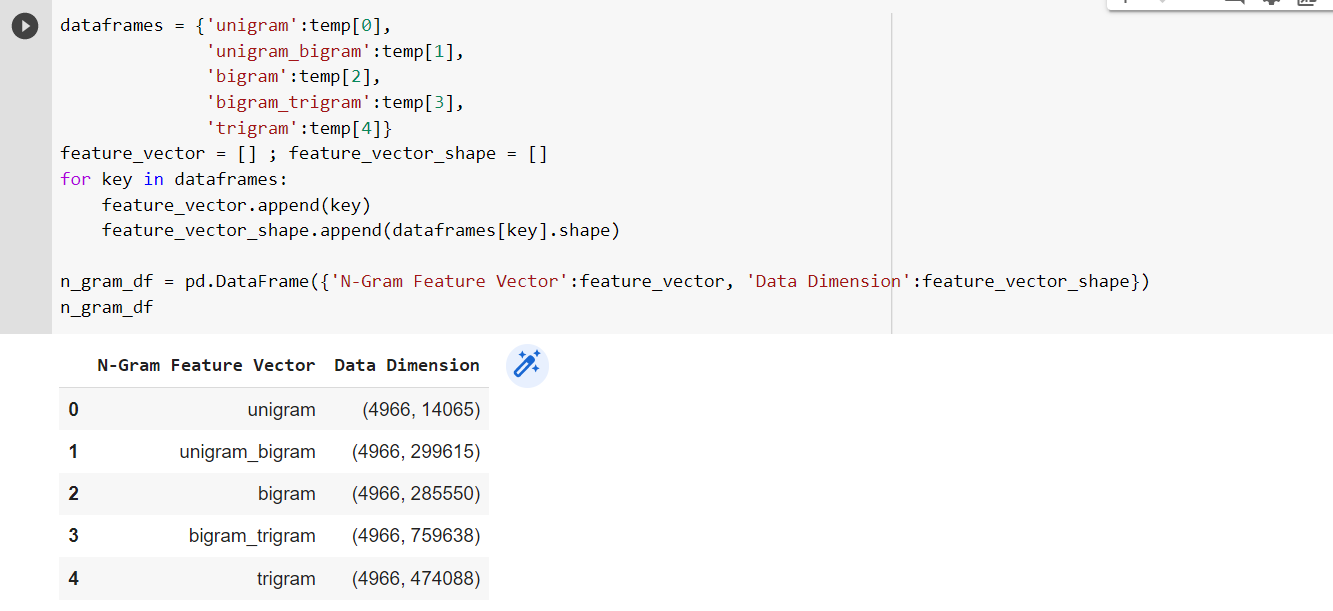


# 3.0. Text N-Gram Feature Extraction

# We will use sklearn class 'CountVectoriser' to extract different n-grams features. In order to do so, the transcription should be converted into a list format, rather than a dataframe. For the purpose of converting into a flat list (i.e., there is no inner list), the function of 'flat\_list' that defined above is used.

# 3.1. Extract 5 Types of N-Gram

#Generate N-gram

3.2. **Dimension of Each Feature Vector**

After the feature extraction process, 5 kinds of n-gram features are extracted. It is interesting to notice that when the number of 'n' getting higher (i.e, n=1:unigram, n=2:bigram, n=3:trigram), there is a higer number of columns. This is due to it is getting harder to find similar features that can be stored in similar column when it has a longer connected words as one featuer. If the feature is unique, it will automatically append additional column to store the feaure.

# to retrieve a unigram feature vector, # to retrieve a bigram feature vector

# to retrieve a trigram feature vector

